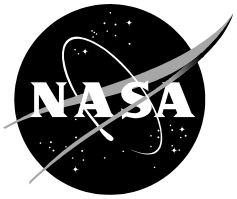
NASA/TP-20220000348



Trajectory Specification Applied to Terminal Airspace

Russell A. Paielli
Ames Research Center, Moffett Field, CA

January 2022

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

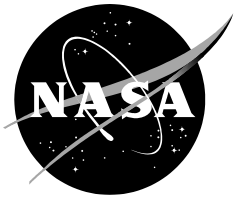
- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TP-20220000348



Trajectory Specification Applied to Terminal Airspace

Russell A. Paielli
Ames Research Center, Moffett Field, CA

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035-1000

January 2022

NASA/TP-20220000348 January 2022

Trajectory Specification Applied to Terminal Airspace

Russell A. Paielli*
NASA Ames Research Center

Summary

Despite major efforts to automate air traffic control (ATC), it is still performed by humans today. The complexity and safety-criticality of ATC makes it very difficult to safely automate, but it must be automated to increase airspace capacity (the density of traffic that can be safely managed) and airport throughput (the number of arrivals and departures that an airport can safely handle in a given period of time) beyond what is possible with human controllers.

This paper presents the Trajectory Specification (TS) concept, which can help to safely automate ATC. TS is a method of specifying aircraft trajectories such that the position at any given time in flight is restricted to a precisely defined bounding space, removing all ambiguity as to where the flight is allowed to be. The bounding space or volume is determined by tolerances relative to a reference trajectory (position as a function of time). The tolerances are dynamic and are based on the aircraft navigation capabilities and the traffic situation. The tolerances can be a piecewise linear function of time or distance along the route, allowing the tolerances to vary as needed, typically increasing with time for departures and decreasing for arrivals.

A Trajectory Specification Language (TSL) is proposed for communicating trajectories from aircraft to ATC as requests and from ATC to aircraft as assignments. The TS concept requires a new generation of airborne Flight Management Systems (FMS) that understand the TSL and can fly the assigned trajectories, but this paper focuses on the ATC functions and the prototype ATC algorithms and software that were developed to test the TS concept.

Assuming conformance, TS can guarantee safe separation for an arbitrary length of time even in the event of an ATC system or communication outage. It can help to achieve the high level of safety and reliability needed for ATC automation, and it can also reduce the reliance on ATC backup systems for tactical conflict detection and resolution during normal operation.

TS can be applied to any controlled airspace, including enroute, terminal, and urban airspace, but this paper presents algorithms and software for arrival spacing and conflict detection and resolution in the terminal airspace serving a major airport. In a fast-time simulation of a full day of traffic in a major terminal airspace, all conflicts were resolved in near real time, demonstrating the computational feasibility and the preliminary operational feasibility of the TS concept.

This paper is a compilation of previous papers, and it adds significant information that was omitted from those papers due to length limitations. It also updates some of the results of those earlier papers due to algorithm refinements and corrections of minor software errors.

*Aerospace Engineer, Aviation Systems Division, Flight Trajectory Dynamics and Controls Branch (Code AFT), Russ.Paielli@nasa.gov

Contents

1 Introduction

Air traffic control (ATC) is currently performed by human controllers using radar displays of traffic and voice communication with pilots. The number of flights that a controller can reliably manage at one time, however, is substantially less than the number that could safely fly in the airspace with an automated ATC system[1, 2]. Controllers are remarkably reliable overall, but they are human and therefore make mistakes. Over 1,800 operational errors (breaches of minimum required separation officially attributed to controller error) occurred in one year in the US, including 55 serious cases in which "a collision was barely avoided" [3] and collision avoidance systems activated. Automation can reduce human error, but an autonomous ATC system that works for all possible traffic situations and conditions is difficult to design and implement and is even more difficult to verify and validate to the required level of reliability and integrity.

Trajectory Specification (TS) is a proposed far-term enhancement of the Advanced Airspace Concept (AAC) being developed by NASA for automating ATC in both enroute airspace [4, 5, 6] and the terminal airspace around major airports [7, 8]. The Trajectory Specification concept was first published in 2005 [9] and has been updated in more recent publications [10, 11, 12]. It has also been issued a US patent. A similar proposal by others [13] with a more airborne focus followed several years after the first publication on the concept.

The main idea of Trajectory Specification is to explicitly specify the allowed deviation from an assigned reference trajectory so that the aircraft position at any given time in flight is restricted to a precisely defined volume of space. By explicitly specifying dynamic tolerances in the deviation from the assigned reference trajectory in all three axes, TS eliminates all ambiguity in trajectory assignment and can guarantee safe separation between flights, out to the conflict-free time horizon that was computed, if they remain in conformance. The TS concept requires a new generation of onboard Flight Management Systems (FMS) that can understand the trajectory specifications and fly in conformance with them, but the focus of this paper is the ATC algorithms and functionality.

This paper is a compilation and update of four previous journal papers on the TS concept [9, 10, 12, 14]. The objective is to combine those previous papers into a single document and to add some additional material and plots that were omitted in the previous papers due to the length limitations of journal papers. It also updates the results after some refinement of the algorithms and software. For background, a description of the concept is presented in Section II. The application of the concept to arrival spacing is presented in section III, and the application to conflict detection is described in Section IV, followed by the conflict resolution methods in Section V. A fast-time simulation that was used to evaluate the feasibility of the concept is explained in Section VI, followed by conclusions in Section VII. The Trajectory Specification Language (TSL) is covered in the appendix.

2 Trajectory Specification Concept

2.1 Overview and Benefits

Trajectory Specification (TS) is the construction of dynamic clearances in the form of virtual roadways or tubes in the sky using data standards, air/ground data communication, and software to specify the parameters. It is more precise, more continuous, more dynamic, and more flexible than the static published routes and discrete altitude restrictions that are currently used to organize traffic in the terminal airspace controlled by human controllers at Terminal Radar Approach Control (TRACON) facilities. It was made possible by advances in communication, navigation, and surveillance. A Trajectory Specification Language (TSL) has been developed (see appendix) to represent these specifications and to communicate them from air to ground and vice versa [14].

Trajectory specification is based on trajectory prediction. Trajectory prediction can be done by ground-based systems or by the FMS onboard each aircraft, but the latter is usually preferable

because it has more accurate and more detailed information about the aircraft itself and the control settings and schedules to be used. An FMS that is programmed for TS will record the assigned trajectories of other flights in the region, and it will attempt to compute a conflict-free trajectory based on (1) the current flight state, (2) the flight intent, (3) wind data, and (4) the assigned trajectories of other traffic. The FMS will downlink that predicted trajectory to ATC as a request. ATC will then recheck the trajectory for conflicts and modify it, if necessary, to resolve any conflicts or other airspace constraint violations. After guaranteeing that the trajectory is free of conflicts, ATC will uplink it as the assigned trajectory, and the FMS will fly it to the specified tolerances (and other FMSs within radio range will record the assignment).

A route is the vertical projection of a trajectory onto the geodetic surface of the earth. In the TS concept, a route consists of alternating straight (i.e., great circle) segments and circular turn arcs. Any point along the route can be specified by the distance along the route relative to an arbitrary reference point on the route (positive in the direction of flight), and that distance will be referred to as the along-track distance or position. A useful convention for terminal airspace is to define the runway threshold as the zero reference point, so that departure trajectories start at, and arrival trajectories end at, zero along-track distance. Then the along-track position indicates the distance flown from takeoff or the distance yet to be flown to landing.

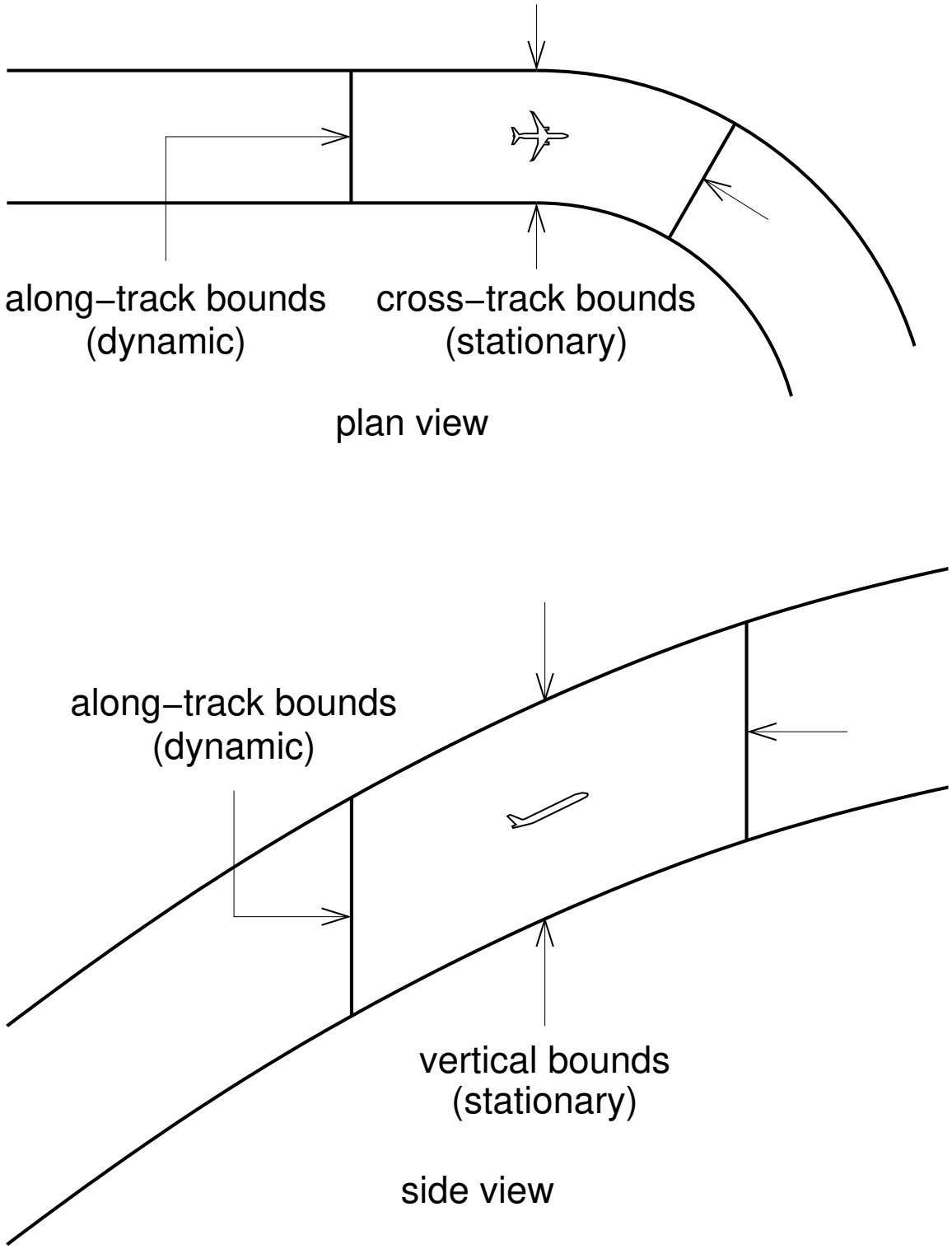
The top diagram of Figure 1 shows an example of a plan view of trajectory bounds at an instant in time. The cross-track bounds are stationary with respect to the earth, and the along-track bounds are dynamic because they move with the aircraft. The lane width is twice the cross-track (lateral) tolerance. The along-track bounds at a given time are vertical rectangles normal to the route direction (which appear as line segments in the plan view) and are defined by the along-track tolerances relative to the reference position at that time. The along-track bounds combine with the cross-track bounds to form a bounding area in the plan view that is a rectangle in the straight segments, an annular area in the turns, or a combination of the two, as shown in the figure.

The bottom diagram of Figure 1 shows an example of a side view of trajectory bounds at an instant in time during a climb. The vertical bounds are stationary and, together with the cross-track bounds, form a stationary bounding tube in three-dimensional space. The along-track bounds combine with the vertical bounds to form a shape with straight vertical sides and curved top and bottom in the longitudinal plane. In level flight, the top and bottom would also be straight. The vertical tolerances in level flight could be ± 100 ft or ± 200 ft, but in climb or descent they would usually be larger, perhaps ± 1000 ft or more, depending on the traffic situation, because altitude is much more difficult to predict and control in climb or descent. The tolerances can vary as a function of along-track distance, but the function itself will be fixed at the time of assignment (or reassignment).

The bounding volume at any given time in flight is a segment or "slice" of a stationary (earth-fixed) bounding tube through which the aircraft is required to fly. Each tube is dynamically constructed for one flight. The vertical cross-sections of the tube normal to the route direction are vertical rectangles, and position along the tube is temporally constrained. (These tubes should not be confused with another tube concept that allows many flights to fly in parallel in a single tube like cars on a freeway.) If one such bounding tube passes over or under another with sufficient vertical separation, then separation is guaranteed as traffic on a freeway is guaranteed to be separated from traffic on a road that passes over or under the freeway. In that case, the trajectories are *statically separated*. If two such bounding tubes intersect or are not spatially separated by the minimum separation standards, then separation must be guaranteed temporally within the tubes, in which case they are *dynamically separated*.

Required Navigation Performance (RNP) [15, 16] is an established technology that limits the allowed cross-track deviation to a specified value for certain published route segments, usually landing approaches. It requires the aircraft to be certified for the particular RNP cross-track tolerance value that is published for the route. TS generalizes RNP to the longitudinal plane by

Figure 1: Top: Plan view of trajectory bounds in the horizontal plane; Bottom: Side view of trajectory bounds in the longitudinal plane



adding vertical and along-track tolerances. Dynamic RNP [17] is a newer technology (not yet available operationally) that allows routes to be created and assigned dynamically, and it also allows discrete altitude constraints and a required time of arrival (RTA). However, DRNP does not specify a continuously bounded trajectory, and it does not allow tolerances that vary along the route, which is important for arrival and departure as will be explained later.

TS is an example of a mathematical concept known as *Differential Inclusion*, which is a generalization of differential equations to apply to finite regions rather than single points in a state space. Because it allows for a finite volume of space rather than a single position at each point in time, TS is an application of differential inclusion, and the bounding tubes introduced below are a special case of trajectory tubes (in the more general sense of a trajectory in state space rather than physical space) [18].

Among the potential benefits of TS is the mitigation of risks that arise in the case of a system outage. Safety must be maintained even if the ATC system or the air/ground data communication goes offline for an extended period of time when traffic density is too high for a human controller to safely take over and manage the traffic. One alternative is to stop any new traffic from entering the affected airspace during the outage (if that can still be communicated somehow) [19]. The traffic will then exit the affected airspace (or land as planned) within approximately 10 to 15 minutes based on the nominally deconflicted trajectories that were previously assigned. Because the deviation from those trajectories is not explicitly bounded, however, conflicts could still arise due to inaccuracies in the wind models, weight, or thrust levels that were used to predict the trajectories.

By explicitly bounding deviation from the reference trajectory in all three axes, TS can guarantee safe separation between flights, out to the conflict-free time horizon that was computed, if they remain in conformance. The conflict-free time horizon will normally be on the order of 15 to 30 minutes, depending mainly on the wind modeling accuracy, which can vary from day to day and even hour to hour. If the ATC system or its data communication link goes offline, the previously deconflicted and assigned trajectories will remain active in the FMS onboard each aircraft to keep the flights safely spaced and separated without the need for any further coordination or communication.

The trajectory will be updated as necessary during the enroute phase of flight, and the time between updates will depend mainly on wind modeling accuracy, but it will perhaps be on the order of 10 to 20 minutes on average. Many of the updates should simply be time shifts to recenter the bounding volume around the current position of the flight (if that can be done without causing a conflict). The trajectory assignment for departures should be computed shortly before takeoff and should extend beyond the terminal airspace, then it should be updated shortly before exiting from terminal airspace into enroute airspace, if necessary. Likewise, a tentative assignment for arrivals should be computed well before entry into terminal airspace, then it should be updated shortly before entry into terminal airspace, if necessary. The trajectory assignment should normally remain unchanged for the entire 10 to 15 minutes that a typical flight spends in the terminal airspace.

As a fundamentally proactive rather than reactive approach to ATC, TS can also provide safety benefits during normal operation. Rather than simply relying on continuous conflict detection and tactical maneuvering when necessary to correct for prediction errors, it facilitates more rigorous, precise, and predictable strategic planning. Tactical backup systems [20, 21] will still be needed, but they should have to intervene less often. The airborne collision avoidance system (ACAS) will also still be maintained as an emergency backup. TS could also facilitate closely spaced parallel approaches or, eventually, formation landing of more than two flights in closely spaced formations to increase runway landing rates.

2.2 Operational Concept

Trajectory Specification (TS) is an extension of trajectory prediction, which will normally be done by the FMS onboard the aircraft. The FMS will take the current flight state, the flight intent, and wind data as inputs and compute a trajectory prediction based on an aircraft performance model. Alternatively, an advanced Electronic Flight Bag (EFB) could potentially be used for this purpose. The intent information includes the route, cruise airspeed and altitude, and possibly other parameters. The FMS will also record and take into account the currently assigned trajectories of other flights in the vicinity to avoid conflicts while computing its own trajectory request. Taking other trajectories into account will greatly increase the probability that the requested trajectory will be free of conflicts and approved by automated ATC without modification. The FMS could use some of the same software components that implement the conflict detection and resolution to be discussed later.

The FMS then downlinks the predicted trajectory to ATC as a request using the TS Language (TSL) [14] mentioned earlier. ATC takes the predicted trajectory as an input and adds default tolerances, which can be constant or a piecewise linear function of along-track position. It then checks the trajectory for conflicts with the current trajectory assignments of other flights and modifies it to resolve conflicts, if necessary, then uplinks it back to the FMS as the assigned trajectory, again using the TSL. ATC will also send the trajectory assignment by landline to all other major ATC facilities through which the flight will fly, including all TRACONS and Air Route Traffic Control Centers (ARTCCs).

If a conflict-free trajectory cannot be found, the flight will be delayed until one can be found (by delaying takeoff time for a departure or by putting an arrival into a holding pattern near the terminal airspace boundary). The pilot (or the FMS, if programmed to do so) can request a new or updated trajectory at any time, and the ATC system should approve it if there are no conflicts or constraint violations. The ATC system will generate a new or updated trajectory, or prompt the FMS to generate one, whenever necessary to resolve a conflict.

The basic operational concept can be summarized as follows:

1. FMS records trajectory assignments from ATC for other flights
2. Pilot enters route and intent data into FMS
3. FMS attempts to compute a deconflicted trajectory prediction
4. FMS downlinks trajectory prediction to ATC as a request
5. ATC assigns tolerances, checks for conflicts and constraint violations
6. ATC modifies trajectory to resolve conflicts and violations if necessary
7. ATC uplinks assigned trajectory with tolerances to FMS
8. FMS flies assigned trajectory to specified tolerances

A possible variation of this operational concept substitutes the Airline Operational Center (AOC) or an Electronic Flight Bag (EFB) for the FMS in all but the last bullet item above. If the AOC is used, the dispatcher or an automated agent would replace the pilot in the second bullet item to enter the desired flight parameters into the AOC trajectory predictor. That possibility may be more feasible before takeoff than during flight, when pilots need to be in control of their flight.

Note that the conflict checks by ground-based ATC are essential for several reasons even if the FMS (or the AOC or EFB) is programmed to receive, and avoid conflicts with, all trajectory assignments for other flights. Firstly, the FMS could miss a trajectory assignment for another flight

if the assigning ATC system is out of radio range at the time of the assignment or the signal is not accurately received for any reason. Secondly, an assignment to another flight could occur while the FMS is computing its own trajectory request, resulting in a potentially dangerous race condition. As an additional benefit, the conflict checks on the ground serve as a backup in case of an error in the FMS conflict detection software.

Trajectory tolerances will depend on the aircraft navigation performance capability and the traffic situation. The navigation capability determines the lower limit of feasible tolerances, and the traffic situation determines the upper limit. The flight operator has no incentive to minimize its own tolerances because that would reduce the airspace volume that it has available, so the default tolerances should come from from a database of minimum feasible tolerances for each aircraft equipage class based on its navigation performance. Alternatively, the operator could be required to use its minimum approved tolerances in its trajectory requests. The ATC system can then increase the tolerances if the traffic scenario allows it, but that is a complex topic that cannot be covered in this paper.

The thrust and airspeed adjustments that are necessary to maintain conformance should be relatively small except in rare cases when the wind model that was used to generate the reference trajectory was grossly in error. Periodic updates can adjust for the accumulated effects of wind errors. If the tailwind is stronger than predicted, for example, the reference trajectory can be shifted in time periodically to re-center the flight, but only if the shift causes no conflict.

The selection of appropriate tolerances for any given aircraft type and traffic situation is too large of a topic to be discussed in detail in this paper. Several issues are involved, including the probability of conformance for a given value of tolerance. That probability should be high, but exactly how high, and how it would be determined, are both open questions that are likely to require further research and analysis. Note, however, that the current lack of explicit tolerances leaves the actual tolerances implicit and therefore ambiguous. Airspace capacity and safety are not likely to be maximized with ambiguous tolerances.

A possible objection to the TS concept is that it continuously constrains the trajectory throughout the entire flight even though potential conflicts may be present over only a small portion of it. Whether this issue would really be a problem in practice is hard to predict, but if it is, several alternatives are available. One alternative is to make the tolerances larger when no other traffic is around and tighter during encounters (using piecewise linear tolerances). Care should be taken, however, to avoid discontinuous or abrupt tightening of tolerances. Another alternative is to stipulate that a breach of tolerances is not enforced if no conflict occurs as a result (and no other flight has to deviate from its assigned trajectory to avoid a conflict).

Another operational issue is non-conformance, which could result from large wind modeling errors, maneuvers to avoid unexpected weather cells, or even pilot error. A flight that is (or is soon expected to be) out of conformance could be handled a few different ways, depending on the situation. If a conflict is not imminent, an attempt can perhaps be made to bring the flight back into conformance, either by adjusting speed or by adjusting the assigned trajectory. If a conflict is predicted to occur within approximately three minutes, then a tactical maneuver can be assigned, consisting of an altitude, heading, and/or speed change. Once the flight is free of conflicts, it can be assigned a new trajectory consistent with its current state (position and velocity). Re-conformance can be a difficult problem for arrivals that need to be reintegrated into a busy arrival stream, but it is beyond the scope of this paper.

The issue of mixed equipage is also beyond the scope of this paper, but a few basic comments are in order. Aircraft that are equipped for TS (i.e., can understand the TSL and fly in conformance with their assigned trajectory) should normally have the "right of way," and unequipped flights should normally be required to maneuver to avoid them when necessary. The Trajectory Specification concept could still be applied to the unequipped flights, but in a slightly different way. Assigned trajectories would not be uplinked to the FMS but would instead be used within the

ATC system to account for the expected level of trajectory prediction error at some estimated level of probability. When an unequipped flight goes out of conformance and is heading for a conflict, a corrective adjustment or maneuver would automatically be sent to it by voice synthesis or by Controller/Pilot Datalink Communication (CPDLC). In fact, this approach could be used as an early application of TS before any aircraft are equipped for it.

Trajectories in terminal airspace will normally be assigned shortly before entry into the airspace, perhaps 1 or 2 minutes before entry for arrivals, and perhaps 30 seconds to a minute before the start of takeoff roll for departures. A tentative trajectory can be computed well before that time, but it should be checked for conflicts with newly assigned (or modified) trajectories and recomputed, if necessary, shortly before it is actually assigned. Once assigned, trajectories should normally not be modified for the entire 10 to 15 minutes that is typically spent in terminal airspace.

In enroute airspace, periodic time shifts or other updates can adjust for the accumulated effects of wind errors, and trajectories can be modified as necessary to accommodate weather changes or to resolve any conflicts that arise. Trajectory modifications during flight should allow a lead time of approximately 1 to 2 minutes for pilot review and acceptance before the new trajectory actually diverges from the old. That lead time can be reduced if the pilot initiates the change request or if the piloting function is automated in the future.

The concepts and algorithms presented in this paper could be used in the near term for trajectory prediction error budgeting in conflict detection and resolution, but the full TS concept requires both a ground-based ATC component and an airborne FMS component. The focus of this paper is the ATC component, for which prototype software has been developed using functional programming methods in the Scala programming language. The resulting software is intended to form the basis for a sound software architecture as well as a starting point for an actual operational implementation. The ATC component is discussed in the next section. The airborne component is not the focus of this paper, but a brief overview in the section following the next section puts it in perspective.

2.3 ATC Functions

In the TS concept, a route is specified as a sequence of waypoints in two-dimensional space (the geodetic surface of the earth) and a turn radius associated with each waypoint. The TS algorithm takes those route parameters and constructs a detailed route representation consisting of alternating straight and turn segments. All turns are tangent-arc or "flyby" turns of constant radius (similar to the "radius to fix" or RF turn leg type in the ARINC 424 navigation standard [22]). If two successive waypoints are too close together for the specified turn radius, the route is geometrically invalid. The route representation constitutes a curvilinear coordinate system in which the coordinates are along-track distance and cross-track position, which can be converted to geodetic or locally level Cartesian coordinates and vice versa.

The use of circular turn arcs simplifies the necessary coordinate transformations but is not absolutely required for the concept. Turns could also perhaps be represented in some other more general mathematical way, but that would be more complicated and will not be discussed in this paper. Note, however, that a turn can always be divided into a series of smaller turns with different radiuses if necessary. The smaller turn segments would alternate with straight segments, but the straight segments can be made arbitrarily short. Such an approach could be used to prevent excessive bank angles for slow aircraft in large turns in high winds, for example. As usual, care must be taken to ensure that the resulting route is geometrically valid (i.e., the length of a straight segment between two turn segments cannot be negative).

The next part of the algorithm is to convert the trajectory data into a set of fast interpolation functions for several flight variables as a function of time and distance along the route. For computationally efficient conflict detection, several flight variables must be accessible in minimal time, including along-track bounds and altitude bounds. In an earlier paper [9], polynomial approxima-

tion was proposed, but in retrospect that was a poor choice because real trajectories usually cannot be modeled well with polynomials unless they are divided into several segments, which adds complexity. In this study, a relatively simple array indexing and interpolation function is constructed as follows.

As explained earlier, the requested trajectory is provided as a series of (time and position) points that are not required to be equally spaced in time (because time steps can be larger for steady-state flight than for turns and altitude transitions). Those points are first converted to route-based coordinates of along-track distance as a function of time. A series of equally spaced times are then generated to span the time interval from the beginning to the end of the reference trajectory, where the constant time step should be somewhere in the range from 2 to 5 seconds (5 seconds was used in this study). The input trajectory data is then "sampled" at those times by finding the input points with the closest bounding times and interpolating between them.

Once the interpolated points with equal time steps are computed, the relevant variable as a function of time or along-track distance is determined by a fast array lookup function. If the start time of the trajectory is t_0, and the time step is Δt, then the array index corresponding to time t is simply $(t - t_0)/\Delta t$. If that value is not an integer (as it usually won't be), the values of the array at the two closest bounding integer indices are interpolated. This procedure provides a fast (constant-time) lookup and interpolation of the relevant flight variables as a function of time or distance.

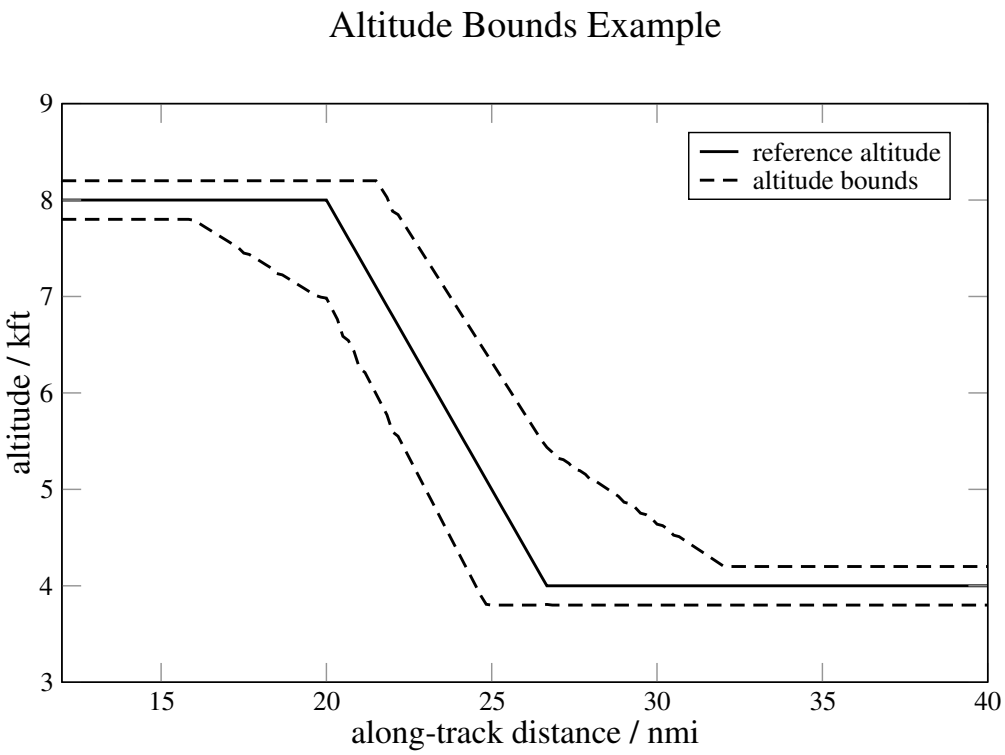
Altitude tolerance during level flight could be ± 100 or ± 200 ft, but during climb or descent it would typically be much larger, on the order of 1000 to 3000 ft. Altitude tolerances in climb and descent can be specified as a constant or as a linear or piecewise linear function of distance along the route. (More general nonlinear functions could also be used, but their usefulness would be unlikely to justify the increased complexity.) The upper altitude tolerance need not be the same as the lower tolerance. The altitude tolerance would typically increase during a climb, but in descent it could decrease as the flight nears final approach. Standard landing systems such as ILS (Instrument Landing System) supersede the assigned trajectory on final approach.

Tolerances should never increase or decrease discontinuously or at a higher rate than the aircraft can follow without causing passenger discomfort. In particular, discontinuities must be avoided at transitions from non-level to level flight and vice versa. Figure 2 shows an example of a simplified reference altitude profile and the resulting altitude bounds as a function of distance along the route. While the reference altitude profile can be plotted as a function of time or distance, note that the altitude bounds cannot properly be plotted as a function of time because the altitude bounds at a particular time also depend on the along-track deviation from the reference trajectory at that time.

Figure 2 is a side view of the stationary, rectangular tube in which the flight is constrained to fly, as discussed earlier. Note the tapered transitions between the level and non-level segments and the cutoff of overshoots. The tapered transitions are at a specified slope in the range of approximately 2 to 3 deg (a slope of 2.5 deg is shown in the figure, although it appears steeper due to very different scales for the axes). Note also that the start of descent is clearly bounded. Lack of such bounds is well known to cause significant problems for automated conflict detection, significantly diminishing airspace capacity [23]. Discretionary descents in particular (in which the pilot is given discretion as to when to start descent) have also caused problems for automated conflict detection, but they can be accurately represented by using larger altitude tolerances.

The cross-track tolerance will typically be constant for long distances as it is for RNP, but it could change discontinuously at the entry into, or the exit from, the terminal airspace, for example. The vertical and along-track tolerances can be specified as a constant or as a linear or piecewise-linear function of distance along the route. This feature allows for tighter tolerances near encounters with other flights or, to put it another way, looser tolerances away from other traffic. It also allows for vertical tolerances to increase during climb as the cumulative effect of trajectory

Figure 2: Simplified example of altitude bounds as a function of distance along route



uncertainty naturally increases. The vertical and along-track tolerances need not be symmetric about the reference position. The along-track tolerance will typically decrease during descent in heavy arrival traffic as the arrival stream compresses on final approach and the normal spacing between flights decreases with time as they get closer to the runway. The vertical and along-track tolerances should never decrease discontinuously or at a higher rate than the aircraft can follow without causing passenger discomfort.

The horizontal bounds form a rectangle in the straight segments, but in turning segments the sides of the rectangle are rounded into arcs as shown in the top diagram of Figure 1. For purposes of computing the separation between bounding spaces, the circular sides can be approximated as piecewise linear (in angular sections of 20 deg or less). This method approximates the horizontal bounding area as a polygon that fully encompasses the turn segments in such a way that the approximated separation will be slightly less than the actual separation but can never be greater. This guarantee is ensured by using an inscribed polyline for the inner arc and a circumscribed polyline for the outer arc.

As mentioned earlier, the altitude tolerance during level flight will be ± 100 ft or perhaps ± 200 ft, but during climb or descent it will typically be larger to accommodate larger uncertainties. Figure 2 shows an example of a side view of the stationary, rectangular tube in which an aircraft is required to fly. The rate of climb and descent are not explicitly bounded, but they must keep the flight within the altitude bounds at all times.

The TS software automatically detects the level segments and applies the default tolerance in those segments. It also applies the tapered transitions between the level and non-level segments and cuts off the altitude overshoots that would otherwise precede or follow a level segment. The tapered transitions are shown at a slope of 2.5 deg but could be varied slightly. The smaller the

taper angle is, the more airspace is reserved. The taper angle should be slightly less than the climb or descent angle to allow enough room for a normal leveloff or a start of climb or descent.

2.4 FMS Functions

The trajectory specification concept will require a new generation of Flight Management Systems (FMSs) and some changes to current flight control methodology. Whereas *any* change to flight control methods and systems requires a substantial effort, including certification, these particular changes are technically feasible and relatively basic. No major technology breakthroughs are needed. The latest generation of FMSs can already conform to specified altitude bounds at a specified position, and they can control to a specified arrival time at a meter fix or runway. The trajectory specification concept generalizes those capabilities to the entire route.

The main function of the airborne component is to receive and parse the assigned trajectory in terms of the Trajectory Specification Language (TSL) (see appendix) and keep the flight in conformance with it. Due to safety criticality and stringent certification requirements, the airborne component will require a major development effort, but the basic ideas are not complicated. The existing flight-control feedback loop could have a low-bandwidth outer loop wrapped around it to monitor for proximity to the bounds and make adjustments as necessary. Vertical speed would be adjusted to maintain vertical conformance, and airspeed would be adjusted to maintain along-track conformance. The pilot can already make these kinds of adjustments through the Mode Control Panel, but they will have to be automated to make the TS concept feasible.

The only required change to lateral flight control is the use of circular turn arcs of constant radius. As mentioned earlier, constant-radius turns are not absolutely necessary for the concept, but they simplify the precise definition of the route, and many FMSs in current use can already fly them [22, 15]. Note that a coordinated turn at constant radius in a wind field requires a varying bank angle. Note also that large turns can be divided into a series of smaller turns, if necessary, to limit the required bank angles. If constant-radius turns are ultimately not considered acceptable, alternatives are possible, but they will not be discussed here.

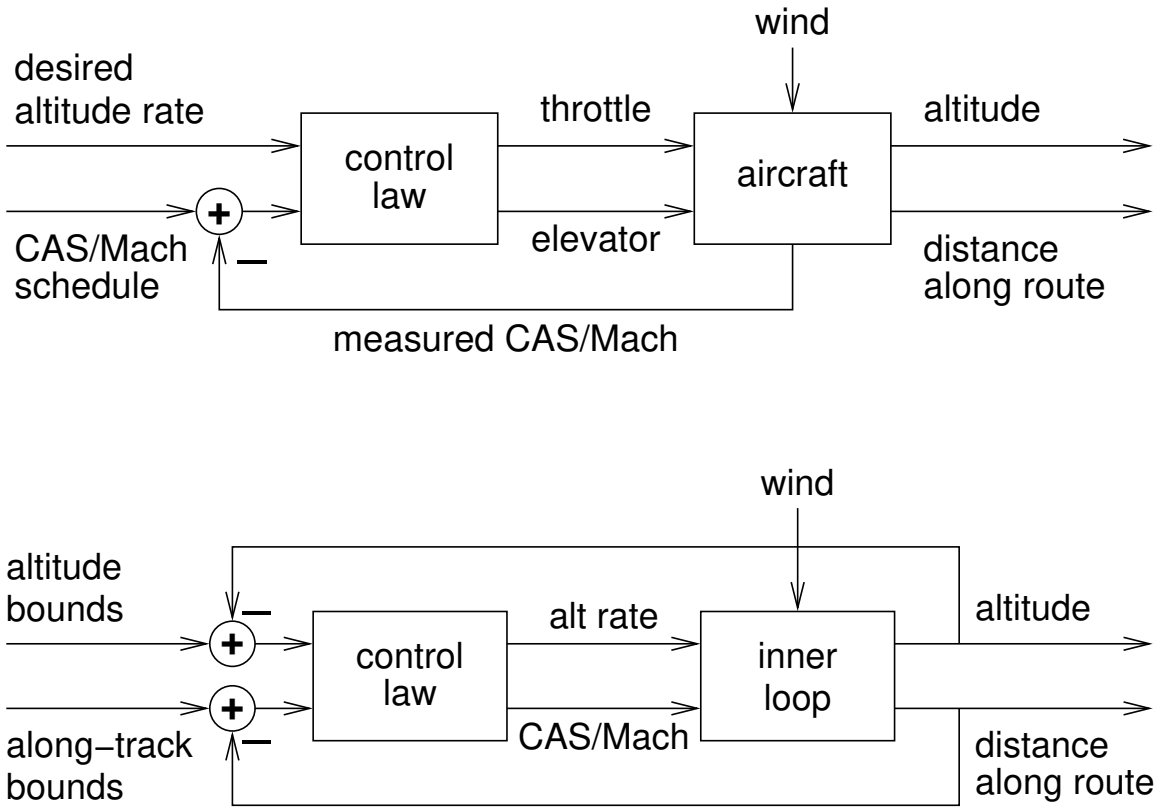
More extensive changes are required for longitudinal flight control to stay within the altitude and along-track bounds in the longitudinal plane as shown in Figure 1. The top diagram of Figure 3 shows a simplified block diagram of aircraft longitudinal control during nonlevel flight. The inputs are the desired airspeed (CAS and/or Mach) and the desired rate of change of altitude; the outputs are the altitude and the distance along track (i.e., along the route). The throttle and elevator settings are the inputs to the engine and airframe as shown in in the figure.

The main feedback variable is the measured airspeed in terms of CAS or Mach (CAS at lower altitudes, including the entire terminal area, or Mach above the CAS/Mach crossover altitude). The main control variable is the elevator angle, which is used to maintain the airspeed schedule by varying the pitch angle and thus the climb or descent rate (trading kinetic and potential energy and causing the flight to climb at a higher rate to slow it down when necessary, for example). For most aircraft, no attempt is made to close the loop on the rate of change of altitude (vertical speed or flightpath angle) in climb or descent (except during final approach). Instead, the throttle is set to some fixed position (or some pre-programmed schedule), and any reasonable resulting altitude change rate is considered acceptable.

In level flight, altitude and altitude rate are also fed back to keep the flight level at the desired altitude, of course. However, even then no attempt is normally made to close the loop on along-track position. Instead, the feedback loop is closed on airspeed (CAS or Mach). Any error in speed control or wind prediction then translates into an error in predicted ground speed, which integrates over time to an error in along-track distance flown.

This open-loop approach to longitudinal flight control is adequate for an individual flight with no other traffic around, but the resulting trajectory prediction uncertainty causes problems for automated conflict detection and ultimately limits airspace capacity. It can also cause problems in

Figure 3: Top: Simplified aircraft longitudinal control; Bottom: Aircraft longitudinal control with a low-bandwidth outer-loop added to bound altitude and along-track distance (the inner loop represents a simplified version of the top figure



the case of ATC system failure as mentioned earlier because the trajectories are not guaranteed to be free of conflicts for any specific length of time.

Errors in the wind data cause errors in the FMS prediction of ground speed and vertical speed, which integrate over time to become errors in altitude and along-track position. If the errors start to approach the allowed tolerances in their respective axes, several kinds of adjustments are possible. One possibility is to adjust the trajectory by shifting it in time to match the current along-route position and/or increasing the tolerances. Such adjustments are allowed only if they do not cause a conflict with the assigned trajectory of any other flight, of course.

Otherwise, adjustments in airspeed and/or thrust may be needed to keep the flight within its assigned tolerances. Airspeed mainly affects the along-track error, and thrust mainly affects the altitude error, but some cross-coupling between the two axes may occur. Those adjustments should be at a relatively low rate to avoid excessive engine transients and passenger discomfort. How close the errors should be allowed to get to their bounds before an airspeed or thrust adjustment is triggered is an important question but is outside the scope of this paper.

The bottom diagram of Figure 3 shows a simplified block diagram of such a control system, where the block labeled "inner loop" represents the control loop shown in the diagram above it. This design "wraps" an outer (low-bandwidth) feedback loop around an existing flight control system, with no major changes to the existing system. Other designs involving more fundamental changes to the flight control system are also possible, but they are beyond the scope of this paper.

3 Arrival Spacing

The schedule of flights arriving at major airports in the US is managed by controllers using an arrival manager called Time-Based Flow Management (TBFM). Originally developed by NASA and known as the Traffic Management Advisor (TMA) [24], TBFM provides a predicted arrival timeline and associated advisories to help controllers maximize airport arrival throughput without overloading runways.

Runway capacity is limited by the wake-vortex spacing requirements and runway occupancy times [25]. The spacing required between arrivals depends on the weight classes of the leading and trailing aircraft. Until recently, the weight classes were designated as small, large, B757, and heavy, but now they are categorized as A-F, where A is the largest and F is the smallest. The required spacing ranges from 2.5 to 8 nmi, but in most cases it is 2.5 nmi. The minimum separation standard in the terminal airspace is 3 nmi horizontally or 1,000 ft vertically. Other more complicated airspace rules and constraints also apply [26, 25], but they are not all enforced in this preliminary study (and some may be unnecessary with more automation in the future).

The guidance provided by TBFM is based on imperfect trajectory predictions, and the execution by controllers and pilots is also imperfect. Moreover, TBFM does not detect or account for separation conflicts. Both spacing and separation conflicts still occur, therefore, and controllers are needed to resolve them. Currently, that control tends to be tactical, with controllers issuing speeds, temporary altitude holds, and heading vectors. That tactical control also forces pilots to fly tactically, with the FMS disengaged, manually entering the speeds, altitudes, and headings into the Mode Control Panel. TS can put flights on automatic, precisely specified trajectories that are guaranteed to be free of conflicts all the way through the terminal airspace.

NASA is currently developing a research prototype software system to automate sequencing and conflict resolution in terminal airspace [7]. This system has been shown to resolve virtually all spacing and separation conflicts in fast-time simulations with current traffic levels. While that is an important advancement of the state of the art, this prototype system has not yet been tested in a more realistic environment with trajectory prediction error and pilots in the loop. Trajectory prediction error is not easy to simulate realistically. A Gaussian or uniform distribution typically does not model outliers well, but outliers pose the most difficult safety challenge. If the errors can be bounded, the problem becomes much more manageable. That is essentially what TS does by imposing explicit tolerances.

In general, spacing conflicts are easier to resolve than separation conflicts because spacing is usually done in a single axis, whereas separation is done in three-dimensional space. Fortunately, the resolution of spacing conflicts at the runway threshold tends to also resolve most (but certainly not all) separation conflicts for flights in the same arrival stream. The general strategy, therefore, is to first resolve spacing conflicts at the runway threshold, then resolve the remaining spacing and separation conflicts, as was done in the earlier study [7].

The analysis and results below are based on 150 arrival trajectories to the Dallas/Fort Worth (DFW) and Dallas Love Field (DAL) airports. All runways are independent at those airports, meaning that spacing is not required between flights on final approach to different runways. The trajectories were generated with the Kinematic Trajectory Generator (KTG) [29], which uses aircraft performance data from the Base of Aircraft Data (BADA) [30] developed by Eurocontrol. The trajectories are based on actual routes flown and were generated by running KTG *without conflict resolution* and storing the resulting trajectories for a simulation period of approximately one hour. The following subsections discuss the methods for modifying the trajectories to resolve spacing conflicts by imposing a delay.

3.1 Arrival Delay Methods

Once an ordering of arrivals to each runway is determined, the required spacing between flights is normally realized by applying a delay to the trailing flight of each consecutive pair when necessary. Spacing requirements are based on the time it takes for a wake vortex to decay to an acceptable level, given the weight classes of the leading and trailing aircraft. However, spacing in terms of time is difficult for controllers, so it is converted into a distance that they can see on their radar screen. For ATC automation, spacing is converted back into time to determine the required delay. Trajectory specification allows the exact delay to be determined to realize a given spacing in terms of distance.

As in previous work [7], delay is achieved in this study by first reducing speed, and if the maximum permissible speed reduction is insufficient, various types of "path stretching" are used. The first type of path stretching is an extension of final approach, and the second type is a symmetric path stretch. These delay methods are explained in more detail below. The resulting overall delay algorithm is intended to be an example of how a specified delay can be systematically realized, but it is not the only reasonable way. If more delay is needed, a holding pattern can be used near the entry into the terminal airspace.

The methods for imposing delay in this study are similar to the methods used in previous work [7], but the *algorithms* are significantly different because they work directly with the original trajectory data rather than calling a trajectory generator for each variation of trajectory parameters to be tried. In other words, the trajectory generator is used to generate the original trajectories but is never used again. That difference is important because in the TS concept the original trajectories are supposed to represent downlinked trajectory requests from an FMS, and the ATC system is not expected to have its own copy of the same trajectory predictor that is used in the FMS. If each iteration of trajectory parameters required to realize a given delay required a call from the ATC system to the airborne FMS, that would clearly not be feasible.

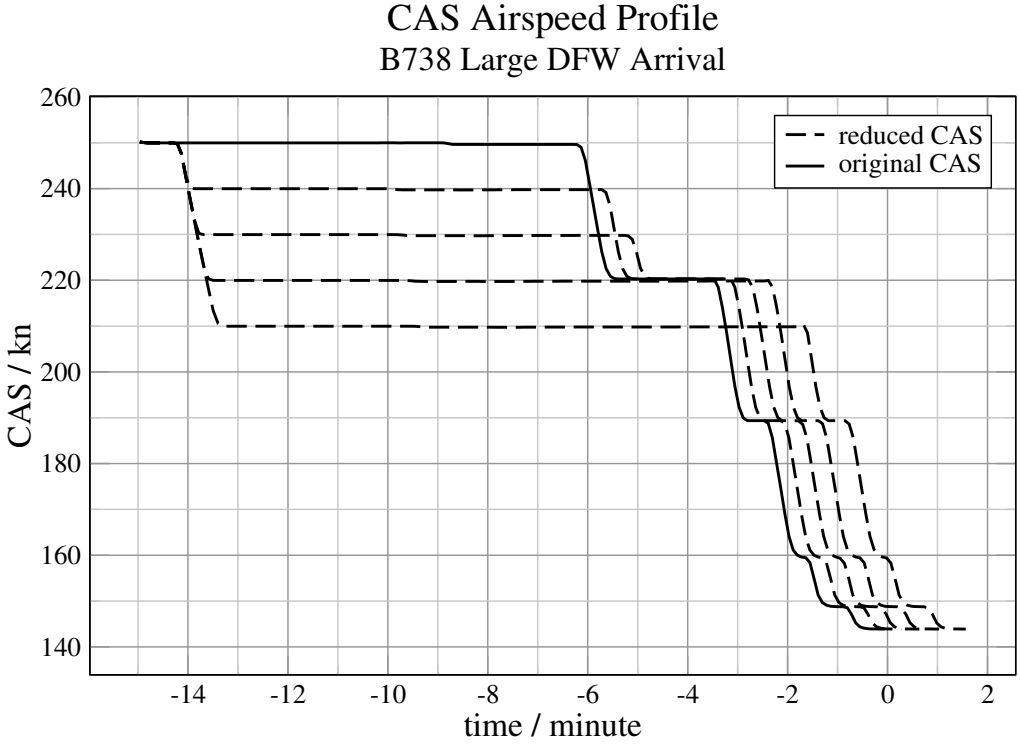
3.1.1 Speed Reduction

As mentioned earlier, the first delay method that is tried for each flight is speed reduction. During descent, speed is normally reduced by steps in Calibrated Airspeed (CAS). At constant CAS, true airspeed (TAS) decreases with altitude during descent. A typical CAS at entry into the terminal airspace is 250 kn, the speed limit in terminal airspace, and that CAS value is normally held constant for several minutes before the step-downs begin to a landing speed typically in the range of 140-150 kn. A delay in the runway arrival time can be realized by reducing the CAS to a lower value during the period of constant CAS. The minimum allowed speed in this study is 1.3 times the nominal stall speed of the aircraft at the given altitude.

Figure 4 shows an example of delays by speed reduction for a Boeing 737-800 (B738) aircraft. In this case, the flight enters the terminal airspace at a CAS of 250 kn as shown by the top (solid) curve and stays at that speed for approximately 8 minutes. The lower (dashed) curves show CAS reductions in steps of 10 kn to a maximum reduction of 40 kn. The speed reduction starts 1.0 min after the entry into the terminal airspace, with a nominal deceleration of 0.06 g. As expected, these curves are very similar in shape to the original speed profile except for the initial speed reduction and the extension to the right on the time line, which represents the delay in arrival time. If plotted verses the along-track distance, the curves would be identical during the decrease in speed.

As mentioned earlier, the tolerances are based on the aircraft navigational capabilities and the current traffic situation. The bounds shown in Figure 5 are examples of altitude and along-track bounds. The along-track tolerances would depend on the proximity of other nearby traffic in the same arrival stream. The altitude tolerances, on the other hand, would more likely depend on departing cross traffic that will pass over or under the arriving flight. Note that the cross-track and altitude bounds are superseded by the localizer and glideslope on final approach.

Figure 4: Example of CAS (calibrated airspeed) profiles for delay by speed reduction

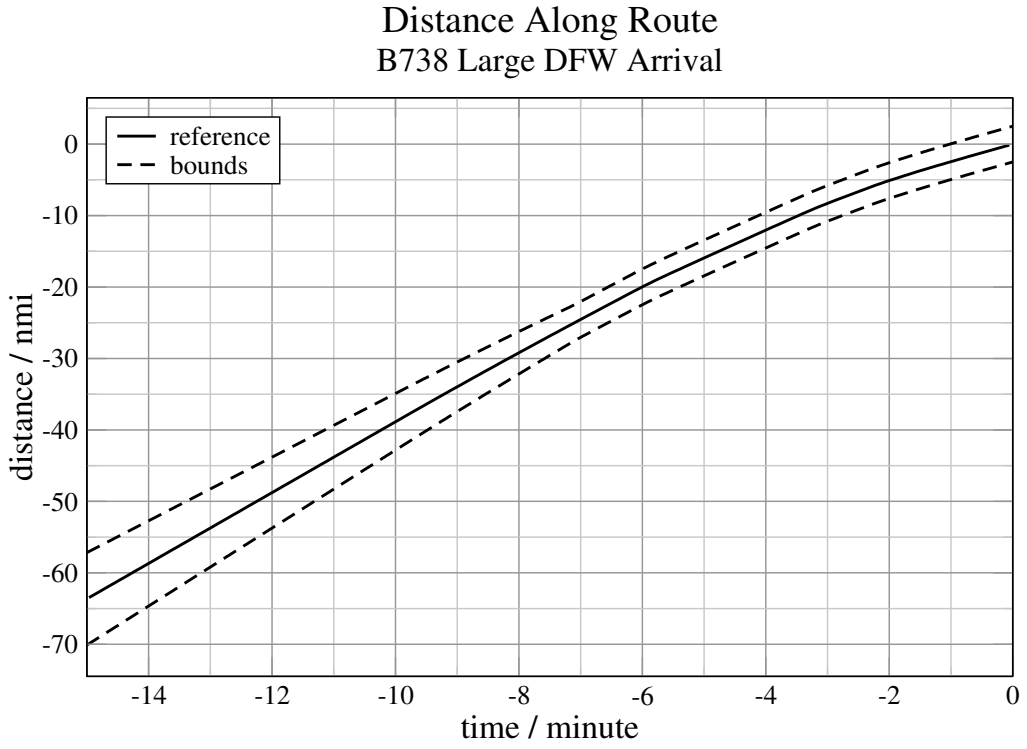


The speed reduction algorithm works as follows. First, the ground speed is calculated by back-differencing the trajectory points and dividing the distance between successive points by the time difference. Note that because the trajectory is a predicted trajectory it contains no measurement noise, so back-differencing is acceptable. Note also that deriving speed directly from position ensures consistency between position and speed. The resulting ground speed is then converted to true airspeed by adding the wind speed in the direction of flight, then the true airspeed is converted to CAS using standard formulas. The CAS in the initial segment of constant CAS is then reduced to the desired value by increasing the time step by the inverse of the ratio of the change in CAS, and the accumulated change in time is tracked. The time for each trajectory point is thus changed while the position remains unchanged. A deceleration limit of 0.06 g is maintained throughout the procedure by limiting the change in CAS at each time step. Once the point in the trajectory is reached where the original CAS is below the new reduced initial CAS, no additional delay is added.

The resulting trajectory then has varying time steps, but it is converted to equal time steps by interpolation as discussed earlier (for fast access to altitude and other variables as a function of time and distance). Because the times of the trajectory points are changed but not the positions, the altitude profile as a function of along-track distance is unchanged, but the altitude profile as a function of time is delayed as necessary.

The speed reduction algorithm determines the resulting delay. What is needed, however, is the inverse of that function, the speed reduction required to realize a specified delay. Figure 6 shows the time delays for one particular flight as a function of the reduced initial value of CAS, with the time of the start of the speed reduction as a parameter. In this study, the CAS reductions were started 1.0 minute after entry into the terminal airspace, as represented by the top curve. The largest delay in this case was slightly less than 1.5 min. As the figure shows, the curves are slightly

Figure 5: Top: Along-track distance and bounds as a function of time; Bottom: altitude profile and bounds as a function of along-track distance



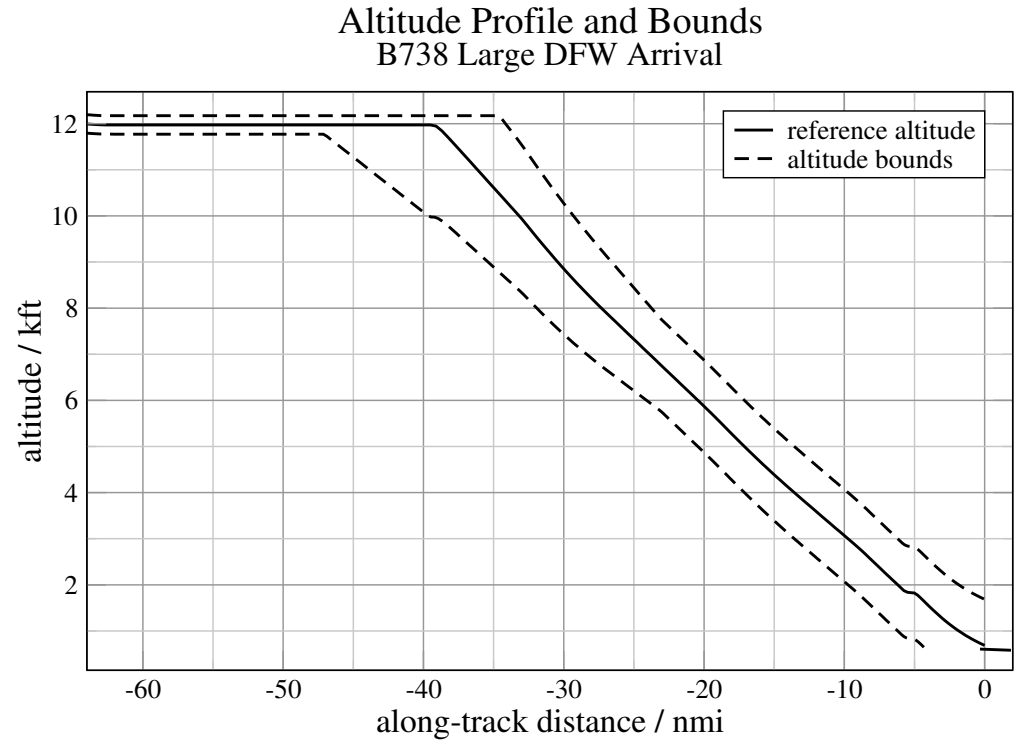
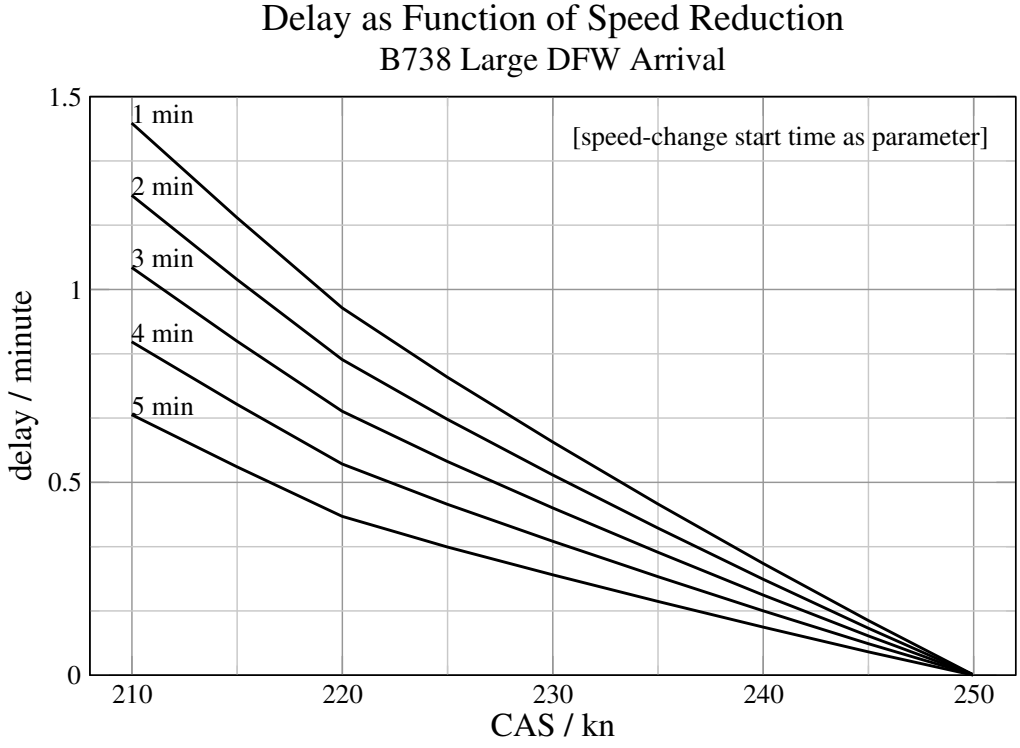


Figure 6: Example of delay as a function of speed reduction



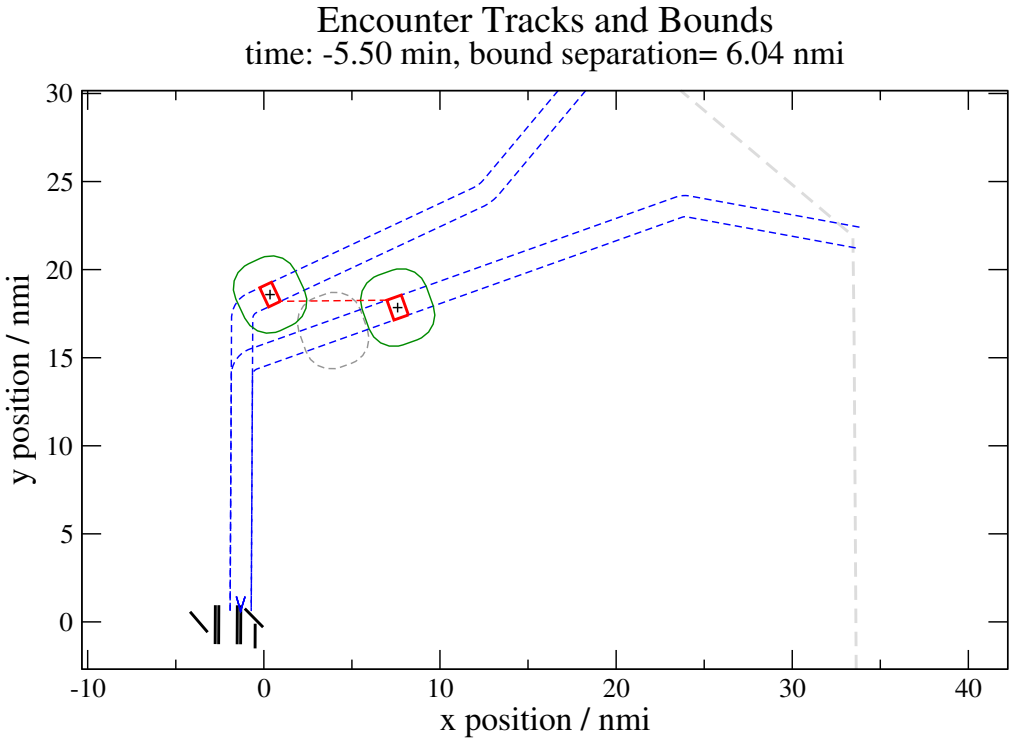
nonlinear but monotonic. The delay accuracy requirement has not yet been rigorously determined, but an accuracy of 1 sec or less is almost certainly more than adequate.

To solve for the CAS required to realize a specified delay, an algorithm was implemented based on the bisection method followed by a final interpolation. This algorithm can achieve arbitrary accuracy at the cost of computation time by simply increasing the number of bisection steps. Five steps were used, followed by an interpolation, and the specified delay was incremented from zero to the maximum achievable delay in steps of 30 sec for each of the 150 arrivals. The largest resulting delay error magnitude was 0.34 sec, and the average time per test was 11 ms on an Intel 2.6 GHz processor (where a test is the computation of one CAS for one delay of one flight). Those levels of accuracy and computation time are more than adequate for use in real time.

Figure 7 shows a snapshot in the horizontal plane of an example of an arrival encounter with spacing by speed reduction. The red rectangles represent the horizontal bounds at a point in time. The along-track tolerances are decreasing linearly to $\pm$ 0.2 nmi at the runway, corresponding to an arrival time window of approximately $\pm$ 5 sec. This value was chosen as an example of a fairly tight but still reasonable tolerance to maximize throughput during an arrival rush, but it may not be appropriate for all aircraft types, and it could be larger in the absence of an arrival rush. If the trailing flight is 5 nmi behind the leader, for example, the back tolerance of the leader and/or the front tolerance of the follower could be increased until the spacing between the bounding areas approaches 3 nmi.

The green ovals around the red rectangles are buffered bounding spaces that extend 1.5 nmi out from the red bounding rectangles. The buffered bounding areas enhance visualization because they touch when the separation between the red bounding rectangles is 3 nmi. The red dashed line connecting the bounding rectangles is the line of minimum separation between bounding spaces,

Figure 7: Snapshot in horizontal plane of an arrival encounter with spacing by speed reduction



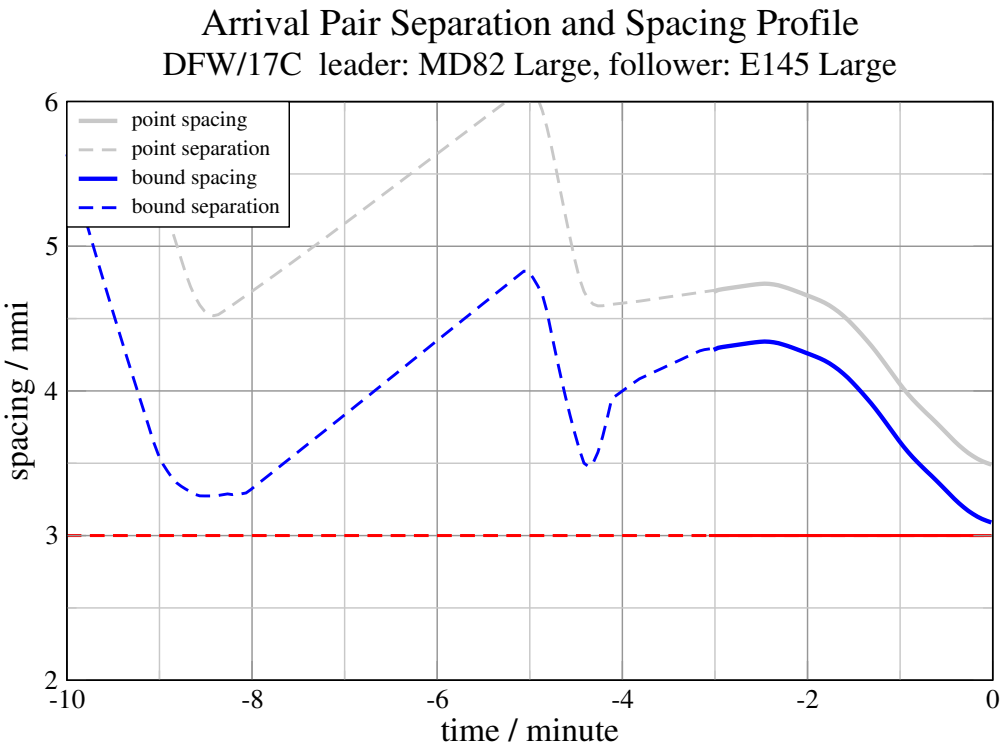
which is 6.04 nmi at this point in time. (Software has been developed to automatically produce a "movie" of an encounter by stacking a series of such plots at regular time increments that can be stepped through manually).

Figure 8 shows the separation and spacing profile as a function of time for the arrival pair shown in Figure 7. The red line at 3 nmi represents the minimum requirement for both separation and spacing for this pair of aircraft. The upper (gray) curve represents the "point" separation and spacing of the reference trajectories, and the lower (blue) curve represents the separation and spacing of the horizontal bounding spaces. For each curve, the dashed segment on the left represents the horizontal separation, and the solid segment on the right represents the spacing while one flight is following directly behind the other on the same path. Because the common intrail segment is straight in this example (on final approach), the spacing is equivalent to the separation. If the common intrail segment extends beyond final approach and has turns, the separation will be smaller than the along-track spacing. The along-track tolerances for each flight in this example are ± 0.2 nmi (approximately ± 5 sec) for each flight on final approach, hence the bound spacing is 0.4 nmi less than the point spacing. An additional buffer of 2 sec is also added between bounding spaces. The large variations in separation are a result of the turns to final. The spacing decreases as usual to a minimum at the runway threshold (at time zero at the right edge of the plot) as the speeds decrease to landing speed.

3.1.2 Extension of Final Approach

When speed reduction is insufficient to realize the required spacing, the next method used is extension of final approach. A waypoint is added to extend the final-approach leg back linearly by a specified distance within the terminal airspace boundary. The control variable is the distance by

Figure 8: Arrival pair separation and spacing profile

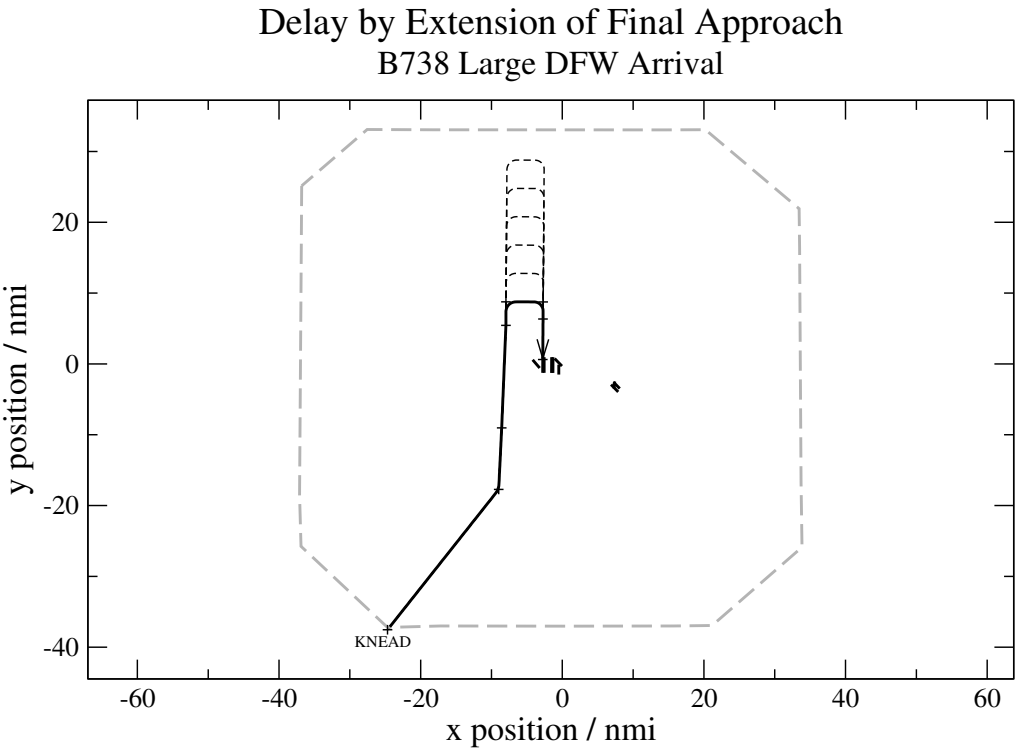


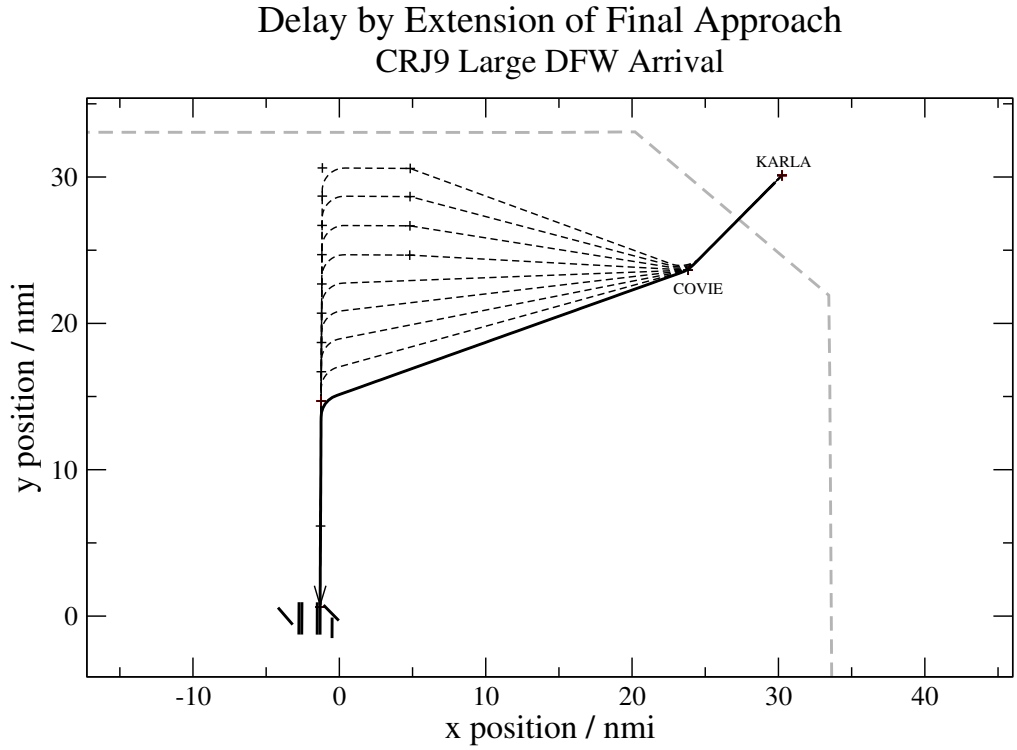
which the waypoint at turn to final is moved back. A base leg is then added if necessary to limit the turn to final approach to a maximum of 90 deg. The top plot of Figure 9 shows an example for an arrival route with a downwind leg, again for a B738 aircraft. The solid line represents the original route, and the dashed lines represent final extensions in steps of 4 nmi. The pattern shown here is the classic "trombone" maneuver. The bottom plot of Figure 9 shows another example with a base leg getting added to prevent the final turn angle from exceeding 90 deg.

In addition to modifying the route geometry as shown in the figures, the algorithm also has to provide a longitudinal profile. The approach taken in this study is based on the simplifying assumption of decoupled lateral and longitudinal dynamics, which is normally a reasonable assumption for commercial passenger airplanes. The original longitudinal profile is superimposed onto the new route starting from the destination runway and going back in time, then a section is added at the start of the route to fill the gap that is left due to the longer route. The altitude as a function of time and distance are therefore identical for the original and the modified trajectory from the runway to back as far as the length of the original trajectory. The same is true for the along-track distance as a function of time. This approach could be refined if necessary to account for some coupling between the lateral and longitudinal dynamics.

Plots of delay as a function of the final extension distance are similar to one curve of Figure 6 and will not be shown. To solve for the final extension distance required to realize a specified delay, an algorithm similar to the one discussed for speed reduction was implemented based on the bisection method followed by a final interpolation. Again, this algorithm can achieve arbitrary accuracy at the cost of computation time by simply increasing the number bisection steps. Each of the 150 arrivals was first reduced in speed by the maximum allowable amount (to 1.3 times the stall speed), then an additional delay by extension of final approach was specified in increments of

Figure 9: Examples of arrival delay by extension of final approach





30 sec. The largest delay error magnitude was 0.06 sec at an average time per test of 16 ms. Again, those levels of accuracy and computation time are more than adequate for use in real time.

3.1.3 Symmetric Path Stretch

When speed reduction and extension of final approach are insufficient to realize the required spacing, the next method used is symmetric path stretching. However, symmetric path stretching is best applied to direct routes, for which extraneous waypoints have been removed. For a direct route, a waypoint is added just inside the terminal airspace boundary to avoid a discontinuous trajectory change at the start of the trajectory, all subsequent waypoints are removed up to the turn to base or final, and the length of the final leg is set to a minimal value of 8 nmi. Ideally, all routes should start as direct routes, and the delay methods discussed above can be applied in sequence as explained above, starting with speed reduction. For various reasons including community noise effects, some direct routes may not be acceptable in the real world, but they will be used in this study.

The plots of Figure 10 show examples of direct routes with maximum extension of final approach followed by a symmetric path stretch, again for a B738 aircraft. The control variable is the perpendicular offset distance from the line segment between the two endpoints, which can go in either direction. The new waypoint must be inside the terminal airspace boundary and is also constrained in two other ways. Firstly, the offset distance was not allowed to exceed 20 nmi or half the distance between the two original waypoints, whichever is smaller. Secondly, the new waypoint is not allowed to cross over the downwind leg because that could interfere with other arrivals to, or departures from, the runway.

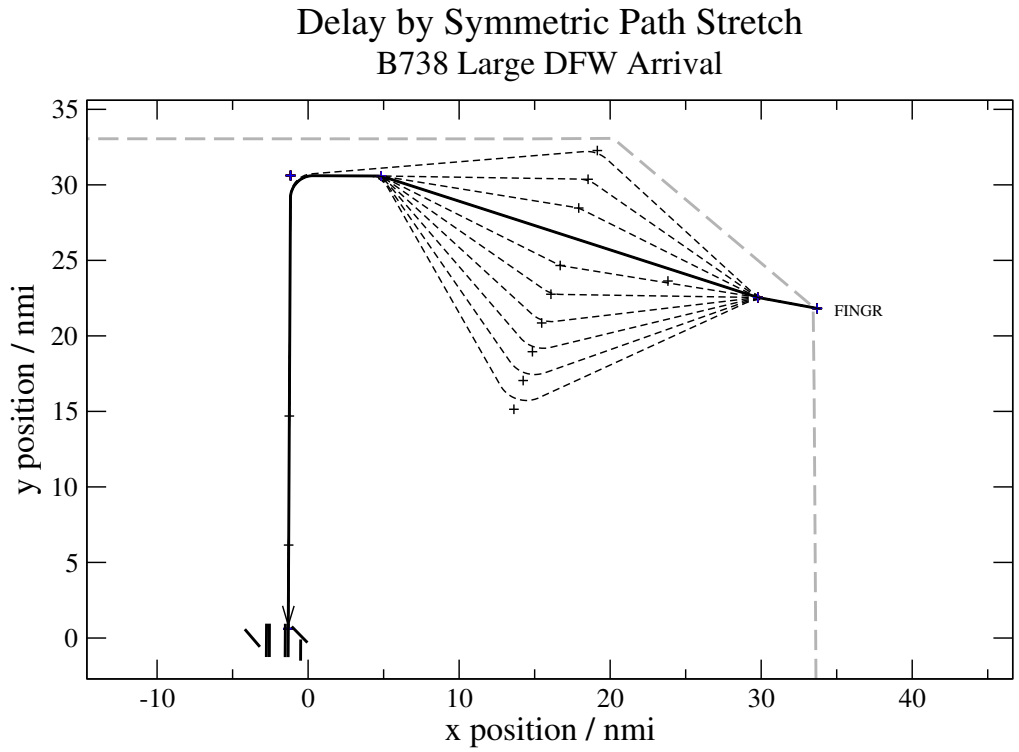
The original longitudinal profile is then superimposed onto the new route using the same algorithm that was discussed in the previous subsection. Also as before, plots of delay as a function of the symmetric path offset distance are similar to one curve of Figure 6 and will not be shown. To solve for the final extension distance required to realize a specified delay, an algorithm similar to the one discussed for speed reduction was implemented based on the bisection method followed by a final interpolation. Each of the 150 arrivals was reduced in speed by the maximum allowable amount as before, and also had its final leg extended to the limit, and then an additional delay by symmetric path stretching was specified in increments of 30 sec. The largest delay error magnitude for the 150 arrivals was 0.04 sec at an average time per test of 6.7 ms. Again, those levels of accuracy and computation time are more than adequate for use in real time.

3.2 Arrival Delay Limits

Figure 11 shows the cumulative distribution of achievable delay at the runway threshold using the methods discussed above for the 150 arrivals used in this study, starting with direct routes to final approach as explained earlier. As can be seen, speed reduction can only guarantee approximately 40 sec of delay, and it can achieve 1.5 minutes of delay for only approximately 40% of arrivals. Extension of final approach is more effective and can achieve approximately 3 min of delay in all cases when added to speed reduction, or 5 min for 60% of cases. Symmetric path stretching adds approximately 1 to 2 min to the achievable delay. If more delay is needed than these methods can provide, a holding pattern near the entry into the terminal airspace can be used for an additional delay of approximately three minutes per loop.

As a final test, the combined delay algorithm (speed reduction followed by extension of final approach, then symmetric path stretching) was tested in delay steps from zero to the delay limits shown in Figure 11 in steps of 1 minute for each of the 150 arrival trajectories converted to a direct route. The largest delay error magnitude was 0.63 sec at an average time per test of 22 ms on an Intel 2.6 GHz processor. As before, these results are more than adequate for real-time application.

Figure 10: Examples of arrival delay by symmetric path stretching



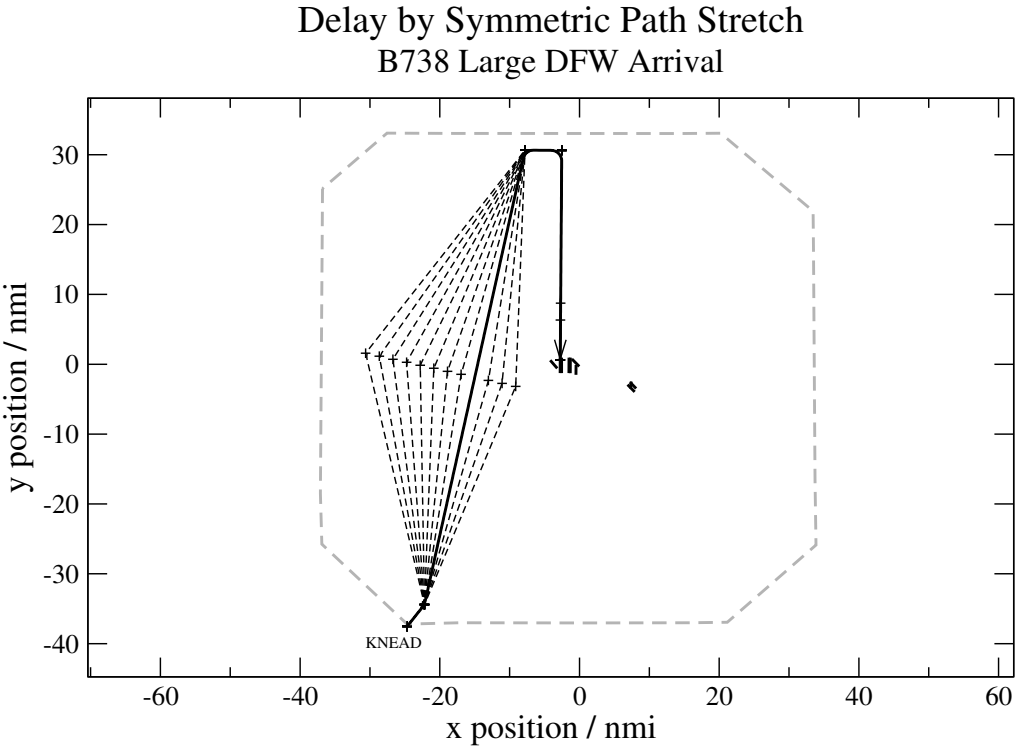
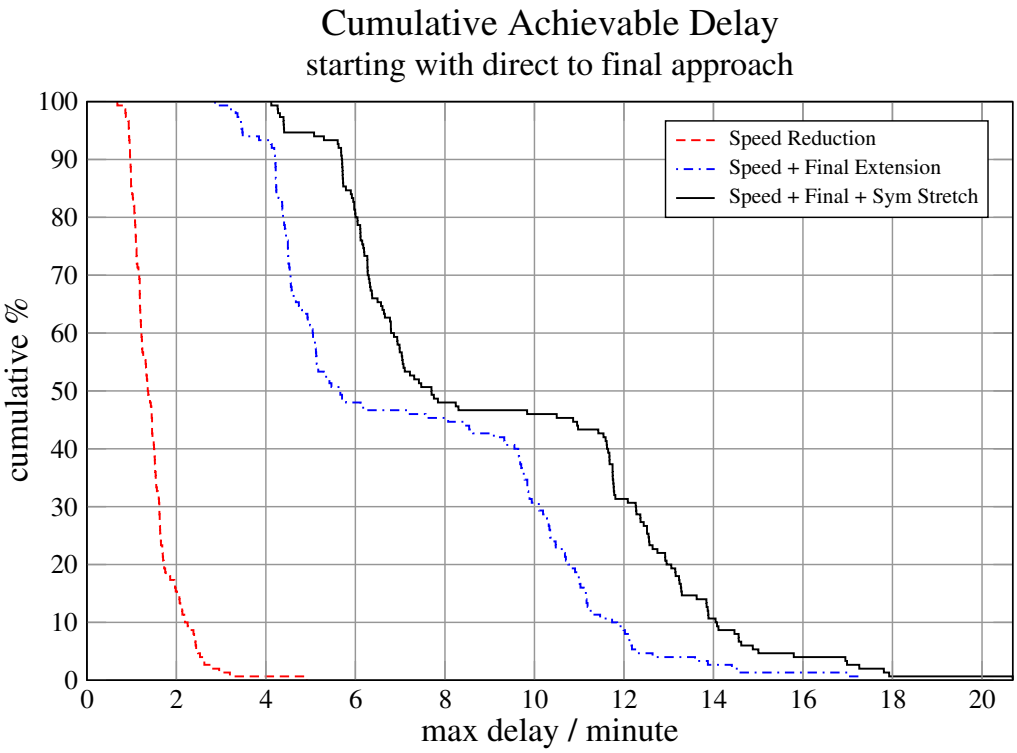


Figure 11: Cumulative distribution of achievable delay for arrivals



4 Conflict Detection

Determining the minimum separation between two trajectories with tolerances requires much more computation than it does without tolerances. Without tolerances, the predicted separation at a given time is a simple calculation of horizontal and vertical separation between two points. If the horizontal separation meets or exceeds the horizontal separation standard of 3 nmi, or if the vertical separation meets or exceeds the vertical separation standard of 1,000 feet, separation is sufficient at that time. With nonzero tolerances, on the other hand, separation is guaranteed to be sufficient at a given time only if every point in the bounding volume for one flight is sufficiently separated with every point in the bounding volume of the other flight. That requires a lot of computation, but a few basic optimization methods can greatly improve computational efficiency.

Alligier et al. [27, 28] developed algorithms for efficiently detecting conflicts between bounding volumes. However, their focus was on tactical maneuvers with uncertain pilot response time, and they make assumptions about the shape of the bounding volumes that do not apply to the TS concept. The horizontal projections of their bounding volumes are assumed to be convex, which is inappropriate for turns. Their algorithm also assumes a single altitude range (minimum and maximum) for each bounding volume, which would be inappropriate for climb and descent.

The reference trajectory can be specified with points that are irregularly spaced in time (because closer spacing is appropriate for turns and altitude transients, whereas larger spacing is appropriate for steady-state flight). However, finding position, altitude, altitude range (upper and lower), or other flight variables as a function of time can be done much faster with array indexing if the points are uniformly spaced in time. One method of improving computational efficiency, therefore, is to convert all trajectory variables that need to be accessed as a function of time (or

Listing 1: Simplified conflict detection algorithm for one trajectory pair

```
 1: HSS = 3 nmi // horizontal separation standard
 2: VSS = 1,000 feet // vertical separation standard

 3: LOS = false // flag to indicate loss of separation
 4: time = (later of trajectory start times)
 5: endTime = (earlier of trajectory end times)
 6: dt = 1 sec // fine time step

 7: while time <= endTime:

 8:     horizSepLowerBound = (calculate horiz separation lower bound at time)

 9:     if horizSepLowerBound > HSS: time += (calculate "safe" time step)

10:     else: // more accurate and time-consuming calculations needed

11:         horizSep = (calculate horiz separation at time)

12:         if horizSep < HSS:
13:             vertSepLowerBound = (calculate vertical sep lower bound)
14:             if (vertSepLowerBound < VSS:
15:               sepRatio = (calculate separation ratio at time)
16:               if sepRatio < 1.0: LOS = true; time = endTime

17:         time += dt
```

along-track distance) into arrays of uniformly spaced points for fast (constant time) lookup by array indexing. The time increment used for this paper was 3 sec. Linear interpolation between adjacent points is also used for better accuracy.

An accurate computation of the separation between bounding spaces is needed only if it is less than or near the minimum required separation. A key to efficient computation, therefore, is to avoid unnecessary computation when the separation is significantly larger than the minimum required, either horizontally or vertically. The following paragraphs explain two kinds of coarse calculations or checks that are used to minimize unnecessary computation; one pertains to spatial separation at a given time, and the other pertains to the time step to the next required spatial separation check. Listing 1 shows simplified, high-level pseudocode of the conflict detection algorithm for one trajectory pair and is explained in the following paragraphs.

Given a pair of trajectories, the separation calculations apply to the overlapping time range of the trajectory pair. As shown in lines 4 and 5 of Listing 1, that time range extends from the later of the two start times to the earlier of the two end times. As indicated by line 8 of Listing 1, the first step is a fast calculation of a lower bound on the horizontal separation between bounding areas, which is done as follows.

The reference position of each trajectory at that start time is determined, and the "pointwise" horizontal separation of the reference trajectories at that time is calculated, neglecting tolerances for now. The along-track and cross-track tolerances will make the horizontal separation of the

bounding spaces less than that pointwise separation. An upper bound is determined for the effect of the tolerances by adding together the cross-track and along-track tolerances of each trajectory at that time, and the result is subtracted from the pointwise separation to obtain a lower bound on the horizontal separation between bounding areas as mentioned above. This method is based on the fact that no point in the horizontal bounding area can be further from the reference position than the sum of the cross-track and along-track tolerances at that time. If the resulting lower bound exceeds the horizontal separation standard of 3 nmi plus some arbitrary buffer (e.g., 2 nmi), sufficient separation is guaranteed at that time, and the exact predicted separation of the bounding areas need not be calculated. The "extra" separation beyond the minimum needed is recorded for use in the next step.

The next method for avoiding unnecessary computation is to skip ahead in prediction time when a quickly calculated lower bound on separation is sufficiently large to guarantee the minimum required separation. This step is shown in line 9 of Listing 1 and is done as follows. An estimate of maximum ground speed is determined for each flight, and the maximum speed for each of the two flights are added to obtain an upper bound on the closing speed, regardless of the relative headings. The "extra" separation at that time, which was calculated in the preceding step as mentioned above, is then divided by this upper bound on closing speed to determine a lower bound on the time at which the horizontal separation could get near or go below the standard. If that time is more than a few seconds ahead, the algorithm skips ahead to that time and repeats the coarse check outlined in the preceding paragraph.

When the upper bound on time to loss of horizontal separation is less than a few seconds, the trajectories are stepped in smaller time increments of one second, and a more accurate separation calculation is performed at each time step. The problem is to determine whether any point in the bounding volume for one flight violates separation standards with any point in the bounding volume for the other flight (at the same time). This calculation is not simply a calculation of the separation between the bounding volumes because the minimum vertical separation can occur at a different point than the minimum horizontal separation.

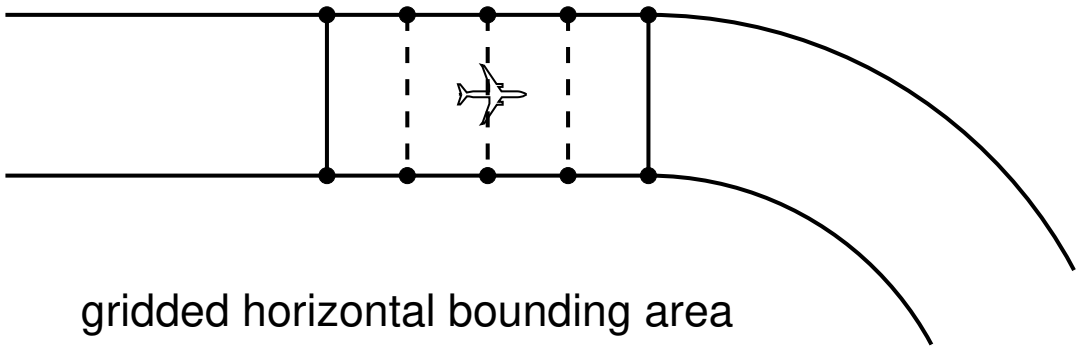
Recall that the horizontal bounding area is a rectangle in the straight segments and is an annular section in the turns as was shown in the top diagram of Figure 1. Standard geometric calculations can be used to compute the separation between the horizontal bounding areas at a given time. This step is shown in line 11 of Listing 1. If the resulting separation is less than or near the standard 3 nmi, no analytical method is known for the combined horizontal and vertical separation calculations. The following numerical method was therefore developed.

In an earlier paper [11], the horizontal bounding area was discretized into a grid of regularly spaced points, but since then a refinement has been found that improves both accuracy and computational efficiency. Because the reference altitude and altitude range (lower and upper bound) is independent of cross-track position, the horizontal bounding areas are discretized into line segments in the cross-track direction as shown in Figure 12. The spacing between the line segments will be discussed later. The line segments are selected in such a way that the front and back along-track bounds are both covered by the selected line segments as shown in the figure (i.e., a smaller increment is used if necessary to cover the ends of the bounding area). The horizontal positions of the end points and the allowed altitude range at those points are then calculated and stored for each line segment.

A coarse vertical separation check is then done as follows. The overall minimum and maximum altitudes are determined for each bounding volume by finding the minimum and maximum of the altitude range for each of its line segments. The vertical separation based on these overall altitude ranges constitutes a lower bound on the vertical separation, and if it is greater than or equal to the standard 1,000 ft, the vertical separation is guaranteed to be sufficient. Otherwise, the following numerical method is used, which is represented on line 13 of Listing 1.

For every possible pair of line segments from one flight to the other (at a given time), the

Figure 12: Horizontal bounding area discretized by cross-track line segments



horizontal and vertical separations are calculated. The vertical separation is the separation between the altitude ranges at that point in space (which is zero if the altitude ranges overlap). A scalar metric called the "separation ratio" is then computed for each possible pair of line segments, as defined below, and the minimum value is recorded. (Determining the separation of two finite line segments in the plane is an interesting problem in itself but will not be discussed here.)

"Separation ratio" is a basic concept that combines vertical and horizontal separations into a single scalar metric. The separation ratio of a pair of points (or line segments) is the maximum of the horizontal and vertical separation ratios, which are defined as follows. The horizontal separation ratio is the horizontal separation divided by the standard 3 nmi. The vertical separation ratio is the vertical separation divided by the standard 1,000 ft. If the vertical separation ratio is 1.0 or more while both flights are fully established in level flight, it is arbitrarily increased to 1.5 to account for the fact that altitude is more stable and predictable in level flight. This convention is useful for distinguishing between higher-risk and lower-risk close encounters, and it is also useful when the target separation ratio is increased slightly above 1.0 as an added safety buffer. A target value of 1.1 is used for conflict resolution later in the paper.

The separation ratio at a time step is the minimum of the separation ratio for each possible pair of points or line segments at that time. The separation ratio for a trajectory pair is the minimum of the separation ratios at each time step of the trajectories. A separation ratio of 1.0 or greater for the entire overlapping time range of the two trajectories means that the minimum required separation is guaranteed as long as both flights are in conformance with their assigned trajectories.

The numerical method explained above can result in a large computational load. For example, if the bounding areas for each of two flights at a given time each have 5 cross-track line segments as shown in Figure 12, the number of pairs is $5 \times 5 = 25$. The positions and altitude ranges are computed only once for each line segment, so the number of position and altitude range computations is 10, but the number of horizontal and vertical separation calculations is 25, which is a significant amount of computation for one instant in time for one pair of trajectories. That computation then needs to be repeated for each time step that the coarse separation checks discussed earlier do not eliminate. That then all needs to be repeated for each pair of trajectories to be checked for conflict. On top of all that, the conflict resolution methods to be discussed later generate many candidate maneuver trajectories that must also be checked for conflict.

The number of line segments for each bounding area is inversely proportional to the discretized spacing between the line segments; hence the computational load for each encounter pair is inversely proportional to the square of that spacing. After some experimentation, a spacing of 0.25 nmi was determined to be a good compromise between accuracy and computational load, given the

horizontal separation requirement of 3 nmi. Using the coarse checks explained above, along with parallel processing in Scala, the separation computations can be done much faster than real time on average, as will be discussed later in the paper. However, the computations can take too long for real time for some difficult conflicts with many candidate resolution maneuvers, as will also be discussed later in the paper. Other opportunities for optimization remain, which can be pursued later as necessary. Note also that, with parallel processing, the computation speed should improve as the number of available processor cores increases in the future.

In terminal airspace, nominal separation standards do not apply in certain situations [26]. For example, if two arrivals are established on final approach on separate, parallel, independent runways, nominal separation standards do not apply. The same is true of two departures on their initial takeoff leg on different parallel independent runways. Nor do nominal separation standards apply to an arrival on final approach and a departure taking off on a different parallel runway. According to official rules [26], a phenomenon called "diverging course" also excuses a violation of nominal separation standards. Diverging course occurs when the intersection of the projected courses is behind one or both flights and the relative course angle is less than 135 deg.

Nominal separation standards may apply for only a portion of the along-track bounding range at a given time. If the along-track range encompasses the turn to final approach, for example, part of the bounding area may be on final approach and part may not be. Therefore the separation requirements must be determined for each pair of line segments in the horizontal bounding areas. To avoid inefficient re-computation, various states are stored with each line segment (in addition to the position and altitude range). These states include flags to indicate whether it is on final approach or initial takeoff, for example.

5 Conflict Resolution

The spacing required between arrivals to, or departures from, a common runway is usually 2.5 or 3 nmi but can be up to 8 nmi, depending on the weight classes of the leading and trailing aircraft. The minimum separation standard in terminal airspace is 3 nmi horizontally or 1,000 ft vertically. In terminal airspace, unlike enroute airspace, conflict resolution is complicated by the constraints associated with arrival spacing requirements. As explained earlier, the approach taken here is to first solve the arrival spacing problem because that also resolves most of the general separation conflicts for arrivals in the same arrival stream to a common runway. Previous studies [7] indicate that approximately 80-90% of arrival conflicts are resolved as a result of arrival spacing.

Many different approaches have been proposed over the years for conflict resolution. In principle, any method that applies to terminal airspace can be applied, but the non-zero tolerances associated with TS further complicate the problem. Previous methods may be adaptable for use with tolerances, but this study adapts methods based on previous work by Erzberger et al. at NASA [7, 8]. Although some refinements were made to those maneuvers, the main contribution of this work is their implementation within the framework of TS with nonzero tolerances.

The resolution methods and algorithms to be presented in this paper are intended to demonstrate feasibility, but for an actual operational system, an advanced development and refinement process with feedback from air traffic controllers will be required that is beyond what can be done in this study. That further development would also address community noise and other issues beyond the scope of this paper.

In the TS concept as applied to terminal airspace, spacing and conflict resolution should normally be done only once for each flight. For departures, it should be done shortly before takeoff; for arrivals, it should be done shortly before crossing the boundary into TRACON airspace. At that point, the flight will be assigned a trajectory that is typically 10 to 15 minutes in duration through the terminal airspace and has no conflict with any previously assigned trajectory. Any change to the assigned trajectory after that point, particularly for arrivals, should be rare and in response

to non-conformance. Because departures are usually less constrained than arrivals, a trajectory modification after takeoff is allowable for a conflict with a new arrival. Cooperative maneuvers involving two or more flights may be useful for some conflicts that are particularly difficult to resolve, but they should be fairly rare and are not used in this paper.

The heuristic approach used here for conflict resolution involves generating and testing many candidate maneuvers in a sort of "scattershot" approach. A more analytical approach may be possible but would be very difficult given the complexity of the problem (including the nonzero tolerances). Several different types of maneuvers are used, including temporary altitude holds, speed reductions, reroutes, takeoff delays, and other maneuver types. For each maneuver type, a list of candidate maneuvers is generated and sorted by the resulting delay. The list is then iterated through, and the first candidate that realizes a target separation ratio (1.1) with a delay less than a given threshold (30 sec) is selected. A separation ratio of 1 or greater indicates a successful resolution, but a target value of 1.1 was used to add an extra safety buffer. If no maneuver is found with a separation ratio above the target value and a delay below the given threshold, but one or more maneuvers are found that resolve the conflict, the one with the smallest delay is selected.

For efficiency, each maneuver is first checked against the original conflicting flight or flights, then only if no conflict is found it is checked against all other traffic. If no candidate maneuver of a particular type (e.g., temporary altitude hold) is found that resolves the conflict, the candidate that yields the largest separation ratio is stored, and the next maneuver type is tried. If none of the maneuver types resolve the conflict, the candidate that yields the largest separation ratio is selected and a second simultaneous maneuver is tried for the same flight using the first maneuver as the starting trajectory. If the conflict is still unresolved after the second maneuver, the flight is held by delaying takeoff for a departure or by putting an arrival into a holding pattern near the terminal airspace boundary.

The conflict resolution algorithm will always be subject to refinement and modification, and the methods used for arrivals and departures are slightly different, but in general the following maneuver types are tried in the order shown:

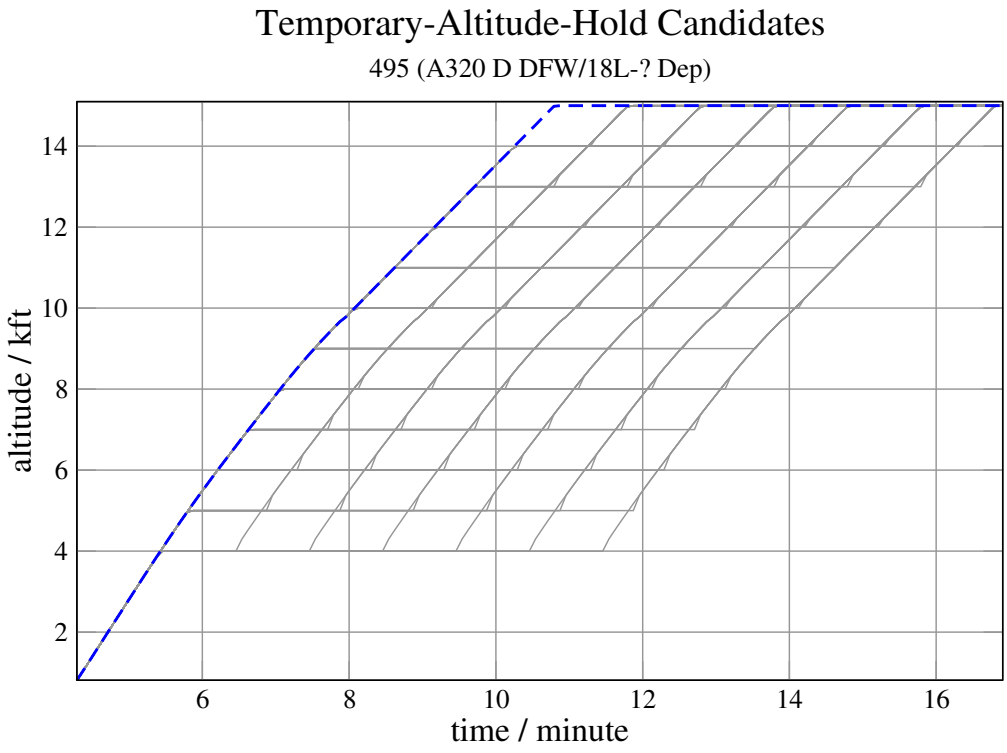
1. takeoff delay (for departures)
2. temporary altitude hold
3. speed reduction
4. reroute
5. other

The order of the maneuver types shown above is the usual order of preference for efficiency (in terms of time or fuel consumption). An altitude hold is usually more efficient than the other maneuver types, for example. The efficiency of an altitude hold is not simple to model, however, and simplified heuristic models are used in this study (based on hold time and deviation from desired altitude). More advanced maneuver efficiency models can be added in future work if necessary. The "other" maneuver category includes various experimental maneuver types that are still being developed and tested for conflicts that are not resolved with the maneuver types listed above. An example will be discussed later.

5.1 Takeoff Delay

The simplest "maneuver" is a takeoff delay, a delay in the brake release at the start of the takeoff roll. Takeoff delays are implemented in this study as simple time shifts of the entire departure trajectory in increments of 15 sec out to a maximum of 4 minutes. In terms of fuel efficiency, a delay on the ground costs virtually nothing and is preferable to a delay in the air. However, if a runway

Figure 13: Temporary-altitude-hold candidates for a departure



departure queue follows behind the delayed departure, the delay will propagate back through the queue. The desirability of this maneuver therefore depends on the state of the departure queue behind the departure to be delayed. If no queue or a short queue follows the departure, a takeoff delay may be the preferred maneuver if it can resolve the conflict. At some airports, departure runways have multiple access ramps for takeoff, in which case the opportunity may exist to reorder the departure sequence to avoid propagating the delay back through the departure queue. The state of the takeoff queues was not considered in this study in selecting the maneuver type, but the propagation of delay back through the queue was accounted for in the delay statistics to be presented later.

5.2 Temporary Altitude Hold

The first maneuver method tried for both arrivals and departures is a temporary altitude hold during climb or descent. This maneuver type is commonly used by controllers today. The flight is simply held at a temporary altitude while on its way to another cleared altitude, until the conflicting flight passes over or under. The candidate altitudes start at the altitude of arrival into, or departure from, the terminal airspace and step down in increments of 1,000 ft, with an arbitrary lower limit of 4,000 ft for this study. For each candidate altitude, the hold time is varied in steps of one minute, up to six minutes, as shown in Figure 13. The increments can be made smaller at the expense of more computation time. The highest altitude and the shortest hold time that realizes the target separation ratio is selected if one is found.

For departures, temporary altitude holds are combined (permuted) with takeoff delays in

increments of 30 seconds up to several minutes to generate more candidate maneuvers.

5.3 Speed Reduction

Speed reduction was also used in this study. Although speed increases are also possible, they tend to be more limited in range and were not considered in this study. The speed reduction applies to the segment of constant CAS (Calibrated Airspeed), typically a few minutes long, which is the initial few minutes in the terminal airspace for arrivals or the final few minutes for departures. An example of speed reduction for arrival delay was shown earlier in Figure 4. The CAS is reduced in steps of 5 knots to a minimum of 1.3 times the stall speed. The speed is held at the reduced value until it drops lower as usual as an arrival gets closer to landing or until a departure exits the terminal airspace.

5.4 Reroute

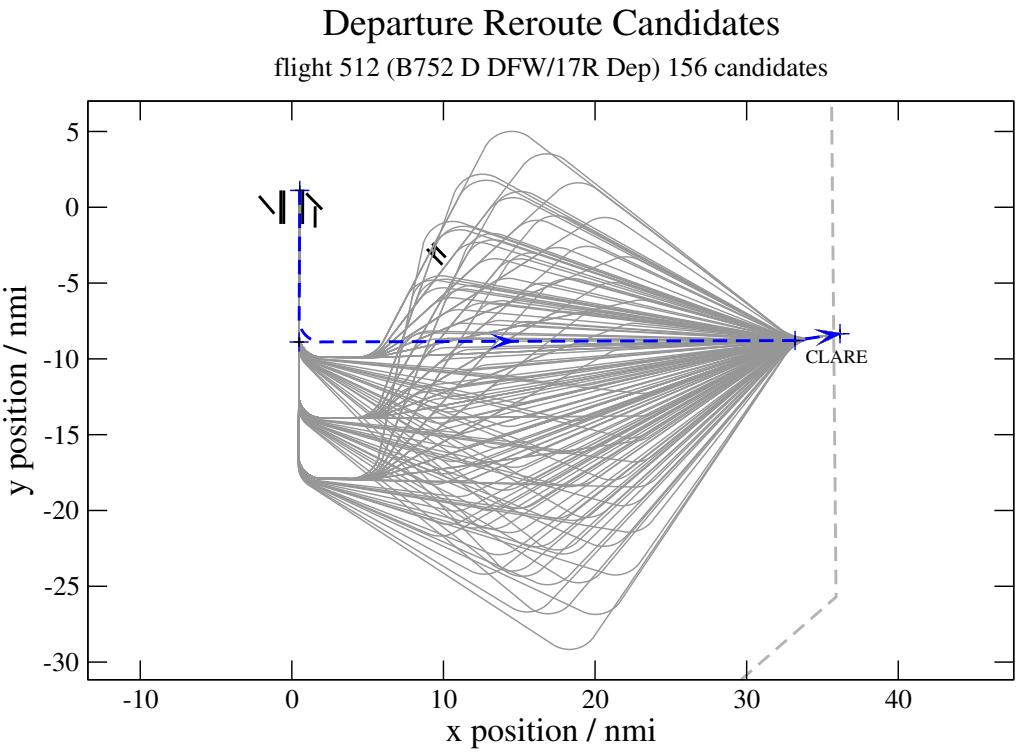
The arrival reroute algorithm starts by generating permutations of three parameters: the turn angle away from the initial direction of arrival into the terminal airspace, the length of the leg in that direction, and the length of the final approach leg. The bottom plot of Figure 14 shows an example of reroute candidates. The turn angle away from the initial arrival leg is varied from 0 to 60 deg in each direction in increments of 20 deg. The length of the leg following that turn is varied from 6 to 16 nmi in increments of 2 nmi. The length of the final approach leg is varied from 8 to 24 nmi in increments of 8 nmi. The increments can be made smaller at the expense of more computation time. Geometrically invalid routes are eliminated (i.e., routes that have turns too large and legs too short to accommodate the specified turn radius). Candidates that go outside of the terminal airspace boundary are also eliminated. A base leg is added, if necessary, to limit the magnitude of the final turn to 90 deg.

The departure reroute algorithm is similar to the arrival reroute algorithm. It also starts by generating permutations of three parameters: the length of the initial takeoff leg, the turn angle away from the initial takeoff leg, and the length of the next leg before turning back to the departure fix (near the terminal airspace boundary). The top plot of Figure 14 shows an example of reroute candidates. The length of the initial takeoff leg is varied from 8 to 24 nmi in increments of 8 nmi. The turn away from the initial takeoff leg is varied from 0 to 60 deg in each direction relative to the original turn angle, in increments of 20 deg, with a turn magnitude limit of 90 deg. As before, the increments can be made smaller at the expense of more computation time. The length of the leg following the turn is varied from 6 to 16 nmi in increments of 2 nmi. As before, geometrically invalid routes and candidates that go outside of the terminal airspace boundary are eliminated.

For both arrivals and departures, the candidate trajectories are constructed and sorted by the resulting delay. The algorithm then iterates through the candidates and selects the first one that successfully realizes the target separation ratio if one is found. If no candidate meets or exceeds the target separation ratio, the candidate that realizes the largest time to loss of separation or the largest separation ratio is stored, depending on whether it is the first or second attempt at resolution.

As mentioned above, the "other" maneuver category includes various experimental maneuver types that are still being developed and tested for conflicts that are not resolved with the previously discussed maneuver types. One such method is base-leg extension, in which the length of the base leg is increased. The base leg, if there is one, immediately precedes the final leg and is perpendicular to it. (This maneuver should not to be confused with a "trombone" maneuver, in which the base leg is shifted away from the runway.) Example maneuvers are shown later in the Results section.

Figure 14: Examples of reroute candidates for a departure (top) and an arrival (bottom)



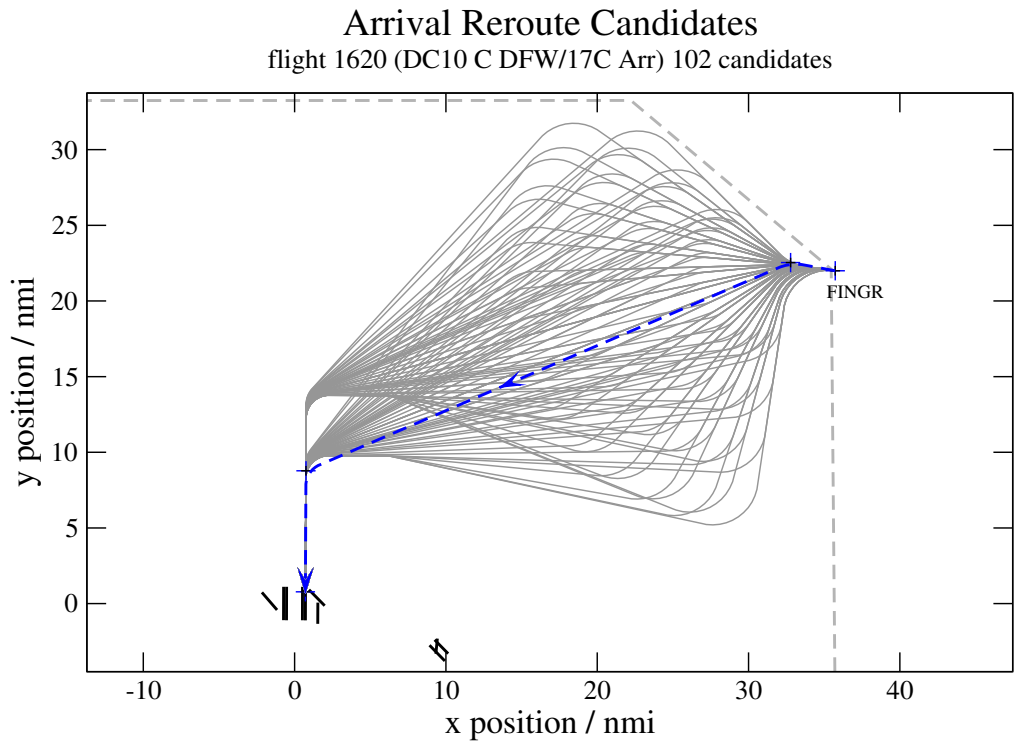
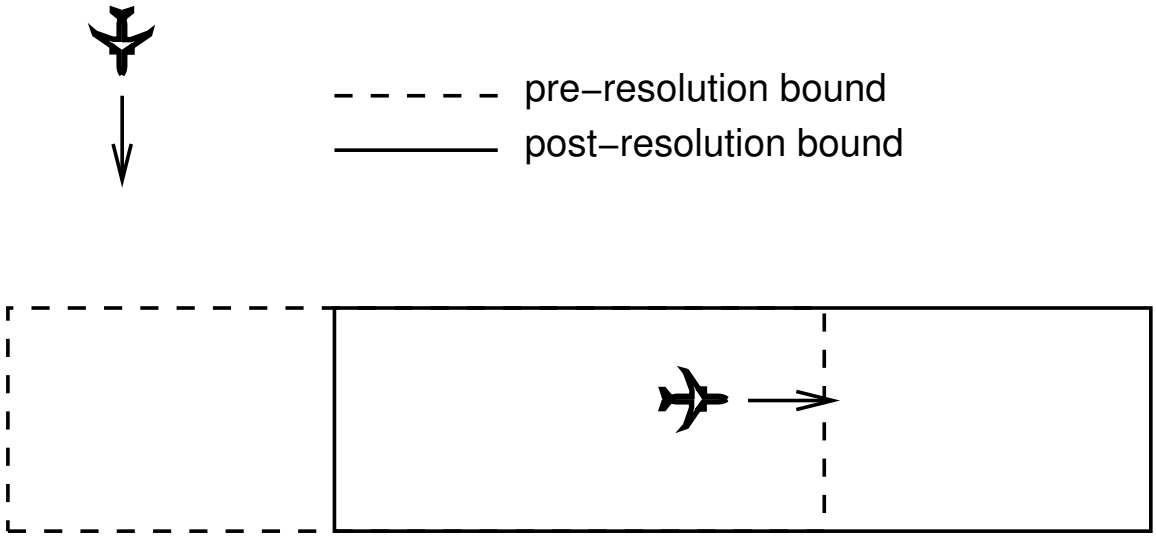


Figure 15: Example of resolution by reframing of bounds



5.5 Resolution By Reframing of Bounds

Some conflicts can be resolved without a maneuver by reframing the bounding space around the flight. Reframing can be done two different ways: (1) by shifting the reference trajectory in time to shift the reference position closer to the actual position; or (2) by temporarily reducing a tolerance near the encounter. Reframing should only be done if the resulting trajectory specification is free of conflicts and spacing violations with all traffic, of course.

Figure 15 shows an example of a flight that is near the front of its bounding rectangle and is in conflict with a flight that will pass behind it. The conflict is resolved in this example by shifting the reference trajectory ahead in time to shift the reference position closer to the actual position. This procedure can be used to recenter the along-track bounds around the flight, but exact recentering is not necessary and may not be the best choice in some cases, depending on the traffic situation. The figure shows the encounter near the time of loss of separation, but the actual resolution time-shift of the reference trajectory would normally be done approximately five to ten minutes earlier, based on the difference between the reference and actual positions at that time. Note that a time shift does not affect the route, so the flight retains the same lateral offset from the route centerline. Note also that this time-shifting method can be used to periodically recenter the along-track bounds around the flight even in the absence of a conflict.

If the time shift shown in Figure 15 were prohibited by a conflict with a flight passing in front, another option would be to temporarily reduce the back tolerance during the encounter. That temporary reduction can be specified by using the piecewise linear functions of along-track distance that were explained earlier. In this case, it would tell the FMS to avoid falling too far behind the reference trajectory during the encounter. The bounding rectangle that results would be similar to the one shown in the figure except that the front bound would remain unchanged. The tolerance values that can be reduced to resolve conflicts include the front and back along-track tolerances and the lower and upper altitude tolerances.

6 Simulation

6.1 Methods

A fast-time simulation was developed to evaluate the Trajectory Specification (TS) concept applied to the terminal airspace controlled by the D10 Terminal Radar Approach Control (TRACON) facility, which serves the Dallas/Fort Worth (DFW) and Dallas Love Field (DAL) airports. Other smaller airports within this airspace were not considered in this study. The main objective was to test the computational feasibility and preliminary operational feasibility of the TS concept applied to the detection and resolution of conflicts in terminal airspace. Trajectory conformance for all flights at all times was assumed for simplicity, eliminating the need for tactical conflict detection and resolution, which is outside the scope of this paper.

Conflict resolution is complicated by the fact that secondary conflicts can arise in the process of resolving a conflict. A previous paper [11] addressed the simplified problem of "pairwise" resolution against only one other trajectory, and this paper addresses general conflict resolution accounting for all traffic. The tests to be discussed in this paper exercise the entire deconfliction process as it would be used in an operational system, including arrival spacing and general conflict resolution for both arrivals and departures. It is worth noting that the interaction of traffic from two different airports in fairly close proximity, as are DFW and DAL, greatly complicates the problem of finding conflict-free trajectories.

As in the earlier section on Arrival Spacing, the starting trajectories used in these simulations were produced with the Kinematic Trajectory Generator (KTG) [29] with no accounting for other traffic (i.e., no deconfliction). Winds were not explicitly modeled in this study, but the effects of wind modeling errors are implicitly accounted for by the trajectory tolerances. (By staying in conformance with the assigned trajectory, the FMS limits the effects of wind modeling errors.) Traditional time-stepped simulation was not necessary because, assuming conformance, the TS concept guarantees separation once a deconflicted trajectory is found and assigned. Nevertheless, an independent test for conflicts between the assigned trajectories was done to verify the validity of the results.

The determination of appropriate trajectory tolerances is a major topic in itself. In general, larger tolerances tend to reduce airspace capacity and increase computation time. For purposes of this paper, however, the default tolerances shown in Table 1 and explained below were used. The cross-track tolerance was set to a constant value of ±0.6 nmi for both arrivals and departures, which corresponds to RNP 0.3. (The RNP cross-track bound is twice the nominal RNP value in units of nmi.) The other default tolerances were set as follows.

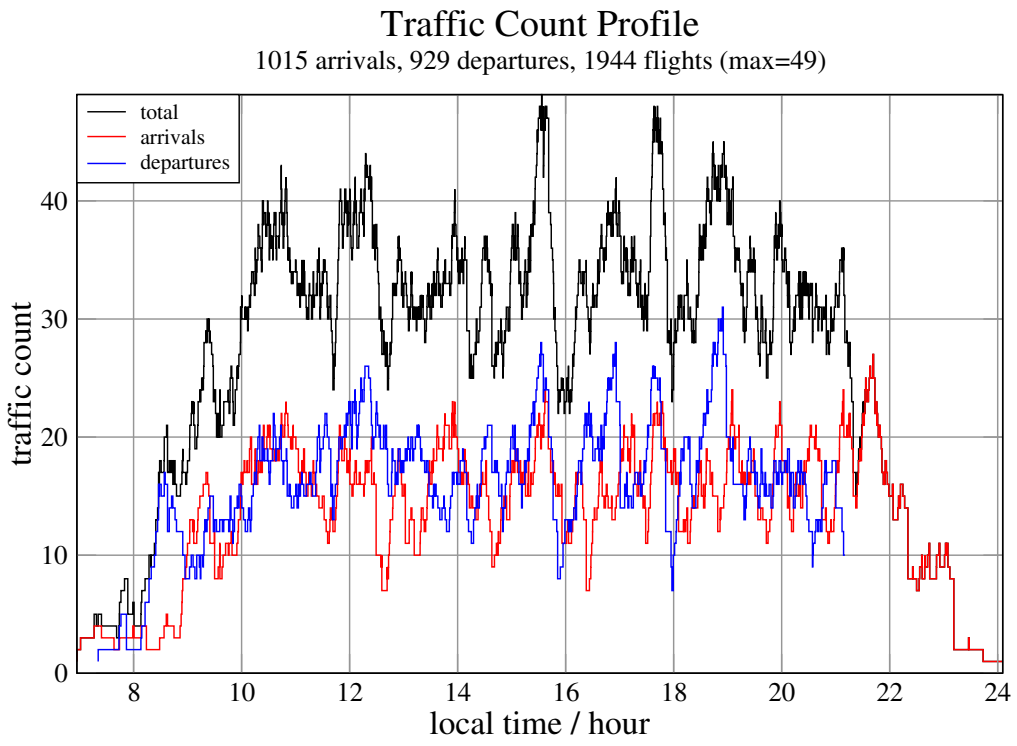
For arrivals, the altitude tolerances were set to a constant value of ±200 ft all the way to final approach, where the glideslope supersedes it. The along-track tolerances for arrivals start at ±0.5 nmi at terminal airspace entry and decrease to ±0.2 nmi near the turn to final approach, corresponding to approximately ±5 sec in terms of arrival time. This decreasing along-track tolerance for arrivals is necessary during arrival rushes to maximize arrival rates at the runway, but it could be relaxed during non-rush periods.

For departures, the altitude tolerances start at ±250 ft at takeoff and increase to ±1000 ft at

Table 1: Default Trajectory Tolerances

	cross-track	vertical		along-track	
		start	end	start	end
arrival	±0.6 nmi	±200 ft	±200 ft	±0.5 nmi	±0.2 nmi
departure	±0.6 nmi	±250 ft	±1000 ft	±0.4 nmi	±1.0 nmi

Figure 16: Simulation traffic count from 6:00 am to midnight



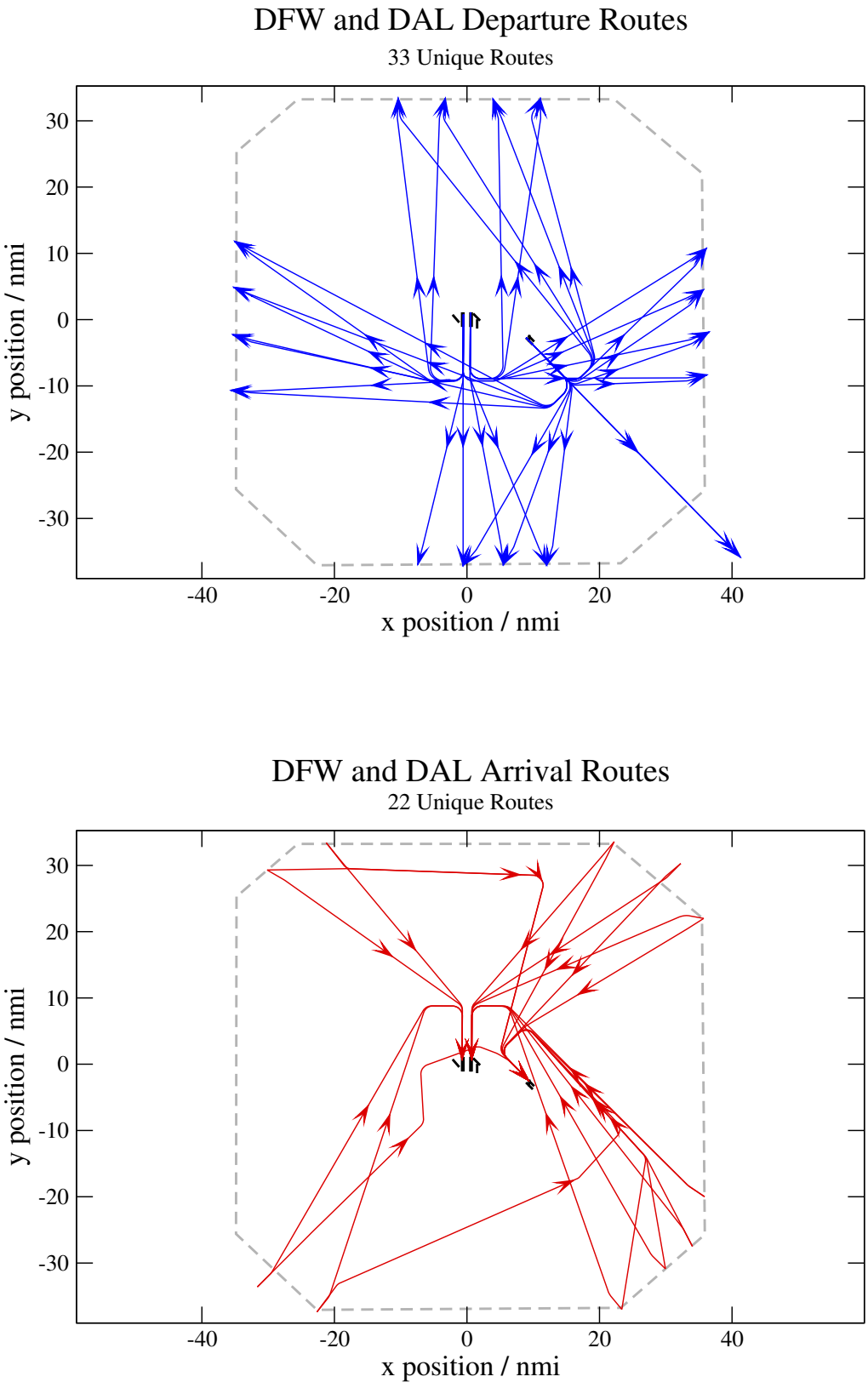
the terminal airspace exit. The along-track tolerances for departures start at ±0.4 nmi and increase to ±1.0 nmi at terminal airspace exit. The increasing tolerance values for departures account for the fact that prediction error typically increases with prediction look-ahead time, and departures are usually unconstrained when they exit the terminal airspace.

These tolerance values are somewhat arbitrary but are considered "realistic" enough for this preliminary feasibility study. The ultimate selection of default tolerance values for each aircraft model is beyond the scope of this paper. They will eventually be refined based on further analysis and testing. The tolerances could be increased for individual flights if doing so does not result in a conflict, but that possibility was not pursued in this study. The tolerances could also be modified based on the evolving traffic situation as the flight progresses through the terminal airspace, but that possibility was not considered here either.

The trajectories used in this test are based on simulated flights in the D10 TRACON airspace for a period of 18 hours, from 6:00 am to midnight, based on recorded flight plans on 4/25/2012. The traffic count throughout the day is plotted in Figure 16, including total flights, departures, and arrivals. The total number of flights was 1944, including 929 departures and 1015 arrivals. A total of 1550 of those flights were to or from DFW, and 394 were to or from DAL. The maximum number of flights active in the airspace at any time was 49. The traffic pattern was the usual "south flow," with DFW runways 17C, 18R, 17L, and 13R used for arrivals, and runways 17R and 18L used for departures. The usual arrival and departure fixes near the terminal airspace boundary were maintained, but routes inside the terminal airspace were modified as follows.

Figure 17 shows the routes that were used in this study. All DFW arrival trajectories were first modified to go direct to final approach, with a base leg added if necessary to prevent the turn to final approach from exceeding 90 deg. Most DAL arrivals were not sent direct to final because

Figure 17: Departure routes (top) and arrival routes (bottom) used in simulation



they would have come too close to the DFW airport and possibly interfered with operations there (by blocking the arrival or departure path of a trajectory to be scheduled later). Departures were also sent directly to their exit fixes, with the equivalent of a base leg added when necessary to keep turn magnitudes from exceeding 120 degrees.

Departures were allowed to climb continuously to 15,000 feet and exit through the top of the terminal airspace rather than leveling out at the usual exit altitudes of 10,000 to 12,000 feet. Eliminating that leveloff in the departure profile improves fuel efficiency and also reduces conflicts with arrivals.

Both departures and arrivals were pre-spaced (by time shifting before the start of the simulation run) for the required runway spacing so that the resulting delay statistics were not adversely affected by runway over-scheduling.

The trajectories were scheduled (and deconflicted, if necessary) in order of their trajectory assignment time, which is defined as follows. For departures, the assignment time is 30 seconds before the takeoff time. For arrivals, the assignment time is 2 minutes before entry into the terminal airspace. Arrivals were then scheduled and deconflicted in order of assignment time. When a large enough gap was found in an arrival sequence and a conflict-free trajectory was found for an arrival that could make use of the gap, the arrival was inserted into the sequence.

Starting with the first trajectory in the sorted list, each trajectory was spaced as required behind the previous arrival or departure on the same runway as discussed earlier. After spacing, each flight was checked for conflicts with all previously assigned trajectories. If the trajectory had no conflict or a successful resolution was found, the resulting deconflicted trajectory was assigned and added to the set of assigned trajectories. If no successful resolution was found, the flight was held for later resolution (to simulate a takeoff delay longer than 4 minutes for departures or a holding pattern of 3 minutes for arrivals). The objective was to resolve all conflicts with as few holds as possible and with acceptable delays.

6.2 Results

This section updates the results from previous papers. Subtle algorithmic refinements and the correction of some software bugs resulted in significantly improved performance in terms of both flight delay (due to resolution maneuvers) and computation time.

For reference, a baseline run was executed with the trajectory tolerances all set to zero, which represents perfect trajectory prediction. As expected, this idealized run was much less challenging than a run with realistic tolerances, both computationally and in terms of conflict resolution performance. In this zero-tolerance run, all conflicts were resolved with no flights held. The mean arrival delay was 18 seconds, and the mean departure delay was 3 seconds. The largest arrival delay was 4:44, and the largest departure delay was 3:00. This zero-tolerance run took approximately 17 minutes to complete, which is 60 times faster than real time on average for the simulated time span of 18 hours.

6.2.1 Resolution Maneuvers

The main test run was then executed with the trajectory tolerances set to the default values shown in Table 1. All conflicts were successfully resolved, and no flights had to be held before a successful resolution was found (an arrival is held by putting it into a holding pattern near the terminal airspace boundary for 3 minutes). Of all flights, 529 (26.6%) required no change to their original trajectory, including 27.0% of departures and 26.2% of arrivals. Of the arrivals, 167 (16.5%) had runway changes, and 65 (6.4%) found conflict-free slots between previously assigned arrivals in existing runway streams. A total of 45 (4.6%) of previously assigned departures were reassigned after takeoff to avoid conflicts with new arrivals.

Table 2: Resolution maneuver type counts (arr: arrival; dep: departure)

maneuvered: / other:	arr / arr	arr / dep	dep / arr	dep / dep	sum
temporary altitude	24	54	31	235	344
speed decrease	255	2	0	0	257
reroute	156	40	5	8	209
new altitude + delay	0	0	4	98	102
temp altitude + delay	0	0	7	95	102
takeoff delay	0	0	0	47	47
new level altitude	1	2	1	17	21
base leg extension	10	5	0	1	15

The resolution maneuver type counts are shown in Table 2. The top row shows the type of the maneuvered flight (arrival or departure), and the second row shows the type of the other (non-maneuvered) flight involved in the conflict. The last column shows the total count for each maneuver type. Plots were automatically generated for each resolution maneuver and examined to verify that the selected maneuvers appeared reasonable.

The most common maneuver type, with 344 cases, was the temporary altitude hold, which was used predominantly for departure/departure conflicts. The second most common maneuver type, with 257 cases, was the speed decrease, which was used almost exclusively for arrival/arrival conflicts. With 209 cases, the reroute was the next most common maneuver type, and it was used mostly for arrivals. Next, with 102 cases, was the new level altitude combined with a takeoff (brake release) delay for departures. (The new level altitude is the altitude at which the departure levels off as it exits the terminal airspace.) A temporary altitude combined with a takeoff delay for departures was also used 102 times. Simple takeoff delays for departures were used 47 times, followed by new level altitudes (without the takeoff delay), which were used 21 times, mostly for departures.

Figure 18 shows a temporary altitude hold for a DFW departure in conflict with a DAL departure. The top plot shows the planview, and the bottom plot shows the altitude profiles. The DFW departure takes off southbound and turns left, while the DAL departure takes off heading southeast and then turns left nearly 180 degrees to a northwest heading. The overlapping dashed green ovals represent the buffered bounding areas at the original (pre-resolution) time of the minimum separation ratio of 0.159. By definition, horizontal separation is the minimum required if and when the buffered bounding areas first come into contact. The temporary altitude for the DAL departure was 9,000 ft for 2.0 minutes, which yielded a minimum separation ratio of 1.28. The solid red rectangles represent the horizontal bounding areas of each flight at the point of minimum separation ratio of the resolved trajectories, and the solid green ovals represent the corresponding buffered bounding areas. (Note that the solid green ovals represent a different time than the dashed green ovals.) The gray ovals represent other traffic at the post-resolution time of minimum separation ratio. In the altitude profiles, the dashed line represents the original trajectory, and the solid lines represent the resolved trajectories. The solid vertical red line represents the time at which the horizontal separation (of bounding areas) drops below 3 nmi, and the dashed green line represents the time at which it goes back above 3 nmi.

Figure 19 shows an example of a departure reroute. A DFW departure takes off to the south and then turns right, conflicting with a DFW arrival heading north on a long downwind leg. The dashed line represents the original departure trajectory, and the dashed green ovals represent the buffered bounding areas at the original time of the minimum separation ratio of 0.300. As shown in the subtitle of the figure, the shortest resolution maneuver found to meet or exceed the target

Figure 18: Example of conflict resolution by temporary altitude hold; Top: planview; Bottom: altitude profile

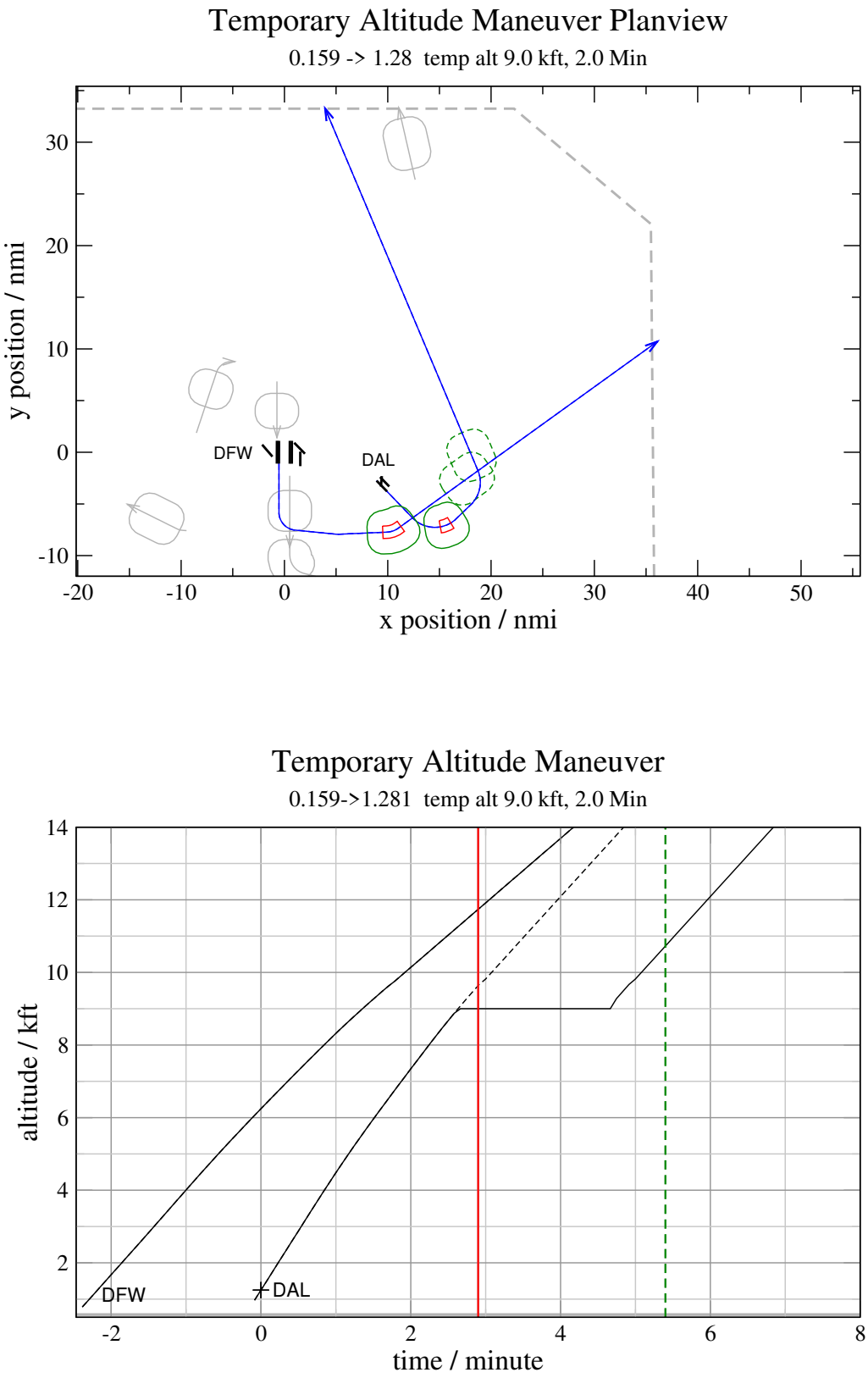
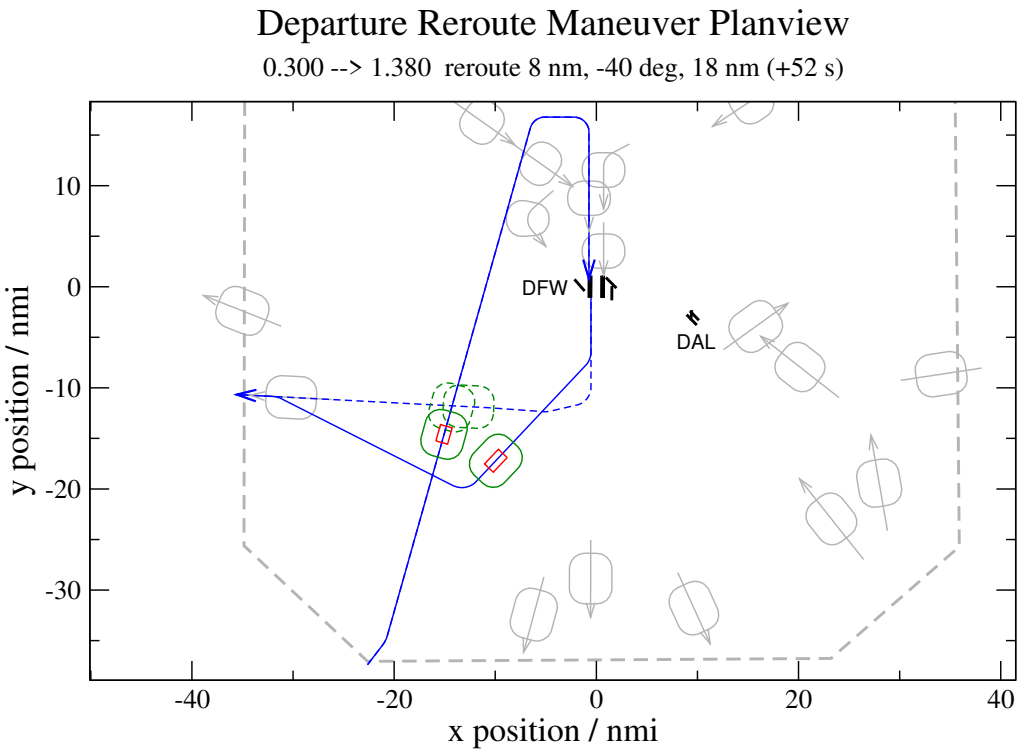


Figure 19: Example of conflict resolution by reroute



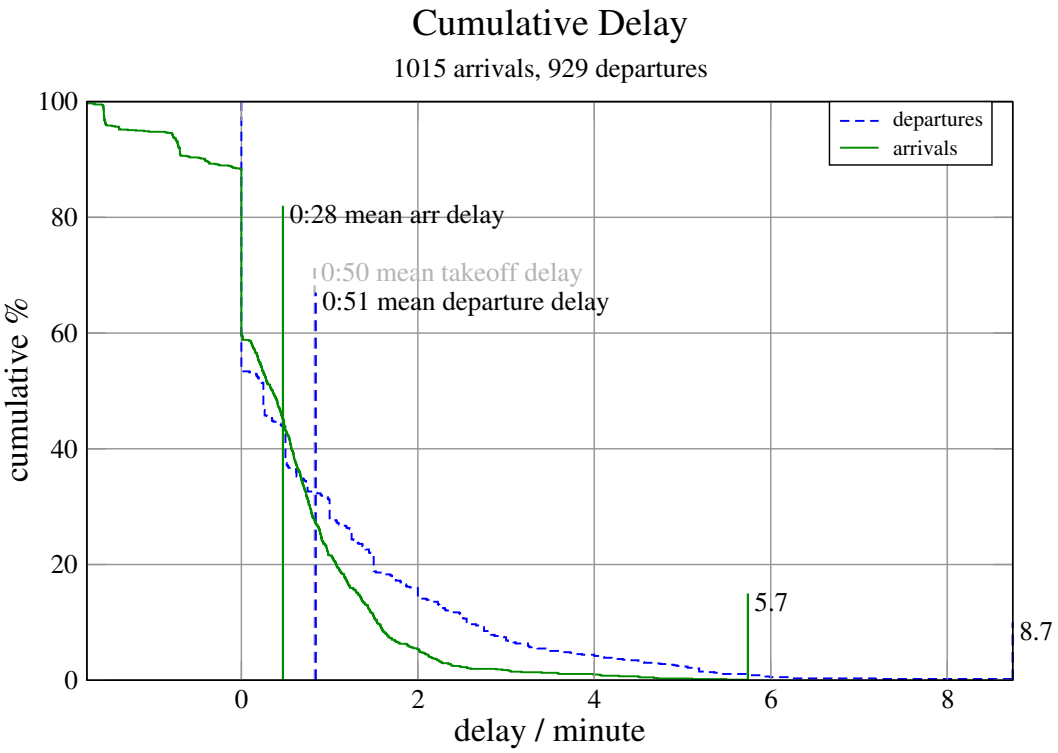
separation ratio of 1.1 had an initial takeoff leg of 8 nmi followed by a turn of 40 deg right to a leg of 18 nmi then a turn toward the original departure fix. As before, the solid red rectangles represent the horizontal bounding areas of each flight at the point of minimum separation ratio after resolution, and the solid green ovals represent the corresponding buffered bounding areas. As before, the gray ovals represent other traffic at the post-resolution time of minimum separation ratio.

6.2.2 Resulting Delays

An important performance metric for conflict resolution is the flight delay that results from the maneuver. Note that the delay results to be presented in this paper do not account for delays due to weather or any demand/capacity imbalances outside of the terminal airspace modeled in this paper. Those externally induced delays are often much larger than the internally induced delays for conflict resolution. As mentioned earlier, in the zero-tolerance baseline run, the mean arrival delay was 18 sec, and the mean departure delay was 3 sec.

Figure 20 shows the cumulative delays for a run with the default tolerances shown in Table 1. This includes delays for runway spacing as well as any delays due to conflict resolution maneuvers. The vertical axis represents the percentage of flights that were delayed by more than the time on the horizontal axis. The mean arrival delay was 0:28, and the mean departure delay was 0:51. These mean delay values are averaged over all flights, whether delayed or not. The mean takeoff delay for departures was 50 seconds, so the total flight delay for departures was almost all takeoff delay (delay of brake release at the start of takeoff roll). As shown on the plot, the maximum arrival delay was 5.7 minutes, and the maximum departure delay was 8.7 minutes. The negative delays are due to arrival runway changes that reduced the flight time.

Figure 20: Cumulative delay plot



The resulting delays depend on the trajectory error tolerances, with larger tolerances resulting in larger delays in general. The default tolerances used in this paper (shown in Table 1) resulted in reasonable delays, but those tolerances may be too tight to be acceptable to the aviation community (note that they apply only in the terminal airspace and would be larger for enroute airspace). The resulting delays for a given level of tolerances could possibly be reduced with more advanced conflict resolution methods and algorithms. Also, future work can be done on allowing the tolerances to be increased as a function of the current and predicted traffic situation and density, which was not done in this study.

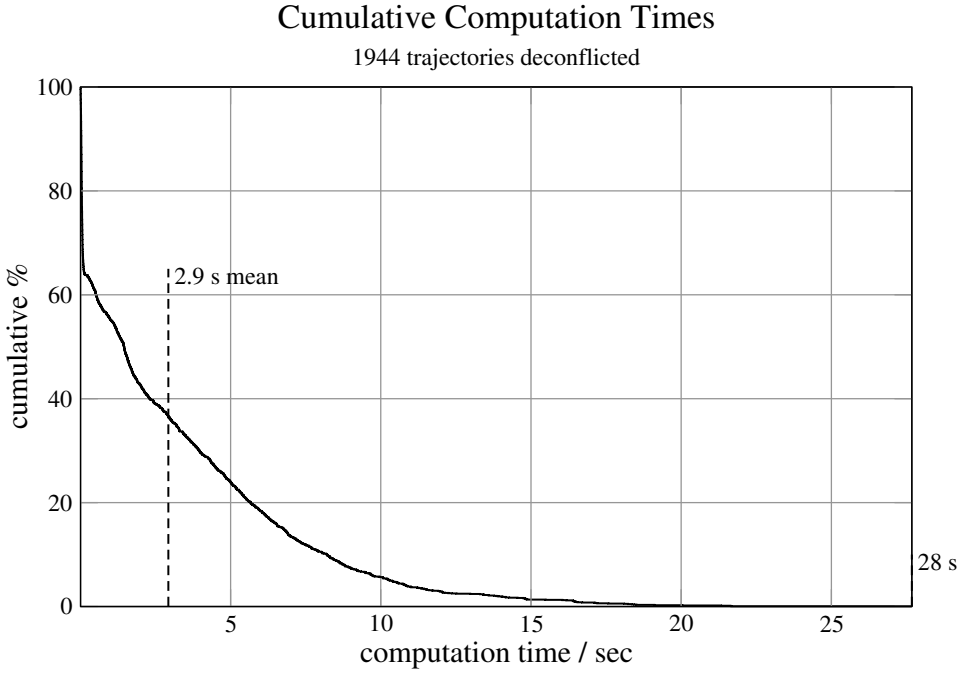
6.2.3 Computation Time

The entire 18 hours of simulated traffic took 120 minutes to process, which is approximately 8.6 times faster than real time on average on an Intel Xeon processor with 32 cores at 2.6 GHz.

Depending on the traffic situation, some conflicts can be much more difficult to resolve than others. Figure 21 shows the cumulative distribution of the times required to deconflict each flight. As shown on the plot, the mean time to find a deconflicted trajectory (including arrival spacing and conflict resolution, if necessary) was 2.9 seconds, and the maximum time taken for any flight was 28 seconds. Five flights required more than 20 seconds to deconflict. Note also that there is no guarantee that a difficult conflict cannot take more than the maximum of 28 seconds that was observed in this particular simulation.

The maximum time required to find a deconflicted trajectory determines how far in advance the search must be started. If deconfliction can take up to 30 sec, for example, then the search must start at least 30 sec plus some buffer time before takeoff or entry into TRACON airspace. A tentative search can be started earlier, but if any assignment for another flight is added or changed

Figure 21: Cumulative resolution computation times



in the meantime and results in a conflict with the tentative trajectory, the search must be done again. Note that for departures, the takeoff time is part of the trajectory, so a new search must be done if the takeoff time is to change. Note also that the search can be done simultaneously by ATC and the FMS in case the FMS (with less computing power) takes too long.

The maximum acceptable time to find a resolved trajectory in real time is not yet established, but a limit of approximately 10 sec seems reasonable, in which case the current TS algorithms and software are not yet fast enough for real time. More work may be needed on algorithmic efficiency. It is worth noting, however, that future computers will have more cores, and a TRACON facility will obviously have more computational resources available for deconflicting trajectories than the engineering workstation that was used for this study.

7 Conclusions

This paper updates the Trajectory Specification (TS) concept as it applies to the terminal airspace serving a major airport. The main idea is that aircraft trajectories are explicitly bounded to a precisely defined volume of space at any given time in flight. TS is a generalization of RNP, adding vertical and along-route bounds to the cross-track bounds that are used in RNP. The tolerances around the reference position are dynamic and will be based on the aircraft navigation capabilities and the traffic situation.

TS can guarantee safe separation for an arbitrary period of time even in the event of an ATC system or communication outage, and it can also reduce the reliance on ATC backup systems for tactical conflict detection and resolution during normal operation. For those reasons, TS could be a key to achieving the high level of safety and reliability needed for ATC automation – automation

that is needed to safely increase airspace capacity and airport throughput beyond what is possible with human controllers.

TS can be applied to any controlled airspace, but this paper focused on terminal airspace, using the D10 TRACON serving DFW and DAL airports as a representative airspace. Algorithms were developed to efficiently detect conflicts, which is much more difficult with nonzero tolerances than it is with the conventional model of zero tolerances. Algorithms were also developed to deconflict traffic, including arrival spacing and general conflict resolution. A fast-time simulation was developed to test the concept, and trajectory conformance was assumed for simplicity, eliminating the need for tactical conflict detection and resolution, which is outside the scope of this paper.

Algorithms were developed to compute and simulate conflict resolution maneuvers. All conflicts were successfully resolved for a full day of traffic using various maneuver types, including temporary altitude hold, speed reduction, rerouting, and takeoff delay. The resolution methods and maneuvers need further development and refinement for actual operational use, but they were sufficient to demonstrate the computational feasibility and the preliminary operational feasibility of the concept. Some conflicts still take longer than acceptable to find a resolution maneuver for in real time, but additional optimization and more processor cores should improve computational performance in the future. Future work is required on selecting appropriate tolerances and ensuring that the tolerances and the resulting delays are acceptable to the aviation community.

8 References

[1] Mercer, J.; Homola, J.R.; Cabrall, C.D.; Martin, L.H.; Morey, S.E.; Gomez, A.N.; Prevot, T.: "Human-Automation Cooperation for Separation Assurance in Future NextGen Environments," *Proceedings of the International Conf. on Human-Computer Interaction in Aerospace (HCI-Aero 2014)*, Santa Clara, CA, 2014. https://doi.org/10.1145/2669592.2669644

[2] Prevot, T.; Homola, J.R.; Martin, L.H.; Mercer, J.; Cabrall, C.D.: "Toward Automated Air Traffic Control—Investigating a Fundamental Paradigm Shift in Human/Systems Interaction," International Journal of Human-Computer Interaction, 28:2, 77-98, Special Issue on NextGen, 2012. https://doi.org/10.1080/10447318.2012.634756

[3] Guzzetti, J.B.: "FAAs Progress and Challenges in Advancing Safety Oversight Initiatives," US Dept. of Transportation, April 16, 2013.

[4] Erzberger, H.: "Automated Conflict Resolution for Air Traffic Control," paper 2006-8.2.1 *International Congress of Aeronautical Sciences* (ICAS) Hamburg, Germany; 3-8 Sept 2006.

[5] Erzberger, H.; Paielli, R.A.: "Concept for Next Generation Air Traffic Control System," *Air Traffic Control Quarterly*, vol. 10, no. 4 (2002), pp 355-378. https://doi.org/10.2514/atcq.10.4.355

[6] Erzberger, H.; Lauderdale, T.A.; Chu, Y.C.: "Automated Conflict Resolution, Arrival Management, and Weather Avoidance for Air Traffic Management," *Journal of Aerospace Engineering*, Volume: 226 issue: 8, pages: 930-949, 2011. https://doi.org/10.1177/0954410011417347

[7] Nikoleris, T.; Erzberger, H.; Paielli, R.A.; Chu, Y.C.: "Autonomous System for Air Traffic Control in Terminal Airspace," *AIAA Aviation Technology, Integration, and Operations (ATIO)* Forum, Atlanta GA, 16-20 June 2014. https://doi.org/10.2514/6.2014-2861

[8] Erzberger, H.; Nikoleris, T.; Paielli, R.A.; Chu, Y.C.: "Algorithms for Control of Arrival and Departure Traffic in Terminal Airspace," *Journal of Aerospace Engineering*, vol. 230, no. 9, pp. 1762-1779, Feb 2016. https://doi.org/10.1177/0954410016629499

[9] Paielli, R.A.: "Trajectory Specification for High-Capacity Air Traffic Control," *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 9, pp. 361-385, Sept 2005. https://doi.org/10.2514/1.12335

[10] Paielli, R.A.: "Trajectory Specification for Terminal Air Traffic: Arrival Spacing," *AIAA Journal of Aerospace Information Systems*, vol. 13, no. 10, pp. 381-392, Oct 2016. https://doi.org/10.2514/1.I010402

[11] Paielli, R.A.: "Trajectory Specification for Terminal Air Traffic: Pairwise Conflict Detection and Resolution," *AIAA Aviation Technology, Integration, and Operations Conf.*, June 5-9 2017.

[12] Paielli, R.A.: "Trajectory Specification for Terminal Airspace: Conflict Detection and Resolution," *AIAA Journal of Air Transportation*, vol. 27, no. 2, April 2019. doi.org/10.2514/1.D0132

[13] Finkelsztein, D.M.; Sturdy, J.L.; Alaverdi, O.; Hochwarth, J.K.: "4D Dynamic Required Navigation Performance, Final Report" NASA/CR2011-217051, Feb 2011.

[14] Paielli, R.A.: "Trajectory Specification Language for Air Traffic Control," *Journal of Advanced Transportation*, Special issue on Emerging Information and Communication Technologies for Traffic Estimation and Control, article ID 7905140, 2018. https://doi.org/10.1155/2018/7905140

[15] RTCA DO-283A: "Minimum Operational Performance Standards for Required Navigation Performance for Area Navigation," Oct 2003.

[16] RTCA DO-236C: "Minimum Aviation System Performance Standards: Required Navigation Performance for Area Navigation," June 2013.

[17] Federal Aviation Administration: "Dynamic Required Navigation Performance: Preliminary Concept of Operations," Version 1.0, RTCA Paper No. 069-14/PMC-1199, March 2014.

[18] Kurzhanski, A.B., Varaiya, Pravin, *Dynamics and Control of Trajectory Tubes: Theory and Computation*, Springer, 2014.

[19] Andrews, J.W.; Erzberger, H.; Welch, J.D.: "Safety Analysis for Advanced Separation Concepts," *Air Traffic Control Quarterly*, vol. 14, no. 1, 2006, pp. 5-24. https://doi.org/10.2514/atcq.14.1.5

[20] Paielli, R.A.: "Evaluation of Tactical Conflict Resolution Algorithms for Enroute Airspace," *AIAA Journal of Aircraft*, vol. 48, no. 1, Jan-Feb 2011, pp. 324-330. https://doi.org/10.2514/1.C031131

[21] Tang, H.; Robinson, J.E.; Denery, D.G.: "Tactical Conflict Detection in Terminal Airspace," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 2, 2011, pp 403-413. https://doi.org/10.2514/1.51898

[22] Aeronautical Radio, Inc.: "Navigation System Database," ARINC Specification 424-20, Dec 2011.

[23] Cone, A.C.; Bowe, A.R.; Lauderdale, T.A.: "Robust Conflict Detection and Resolution around Top of Descent," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conf.*, Indianapolis, IN, 17-19 Sept 2012. https://doi.org/10.2514/6.2012-5644

[24] Erzberger, H.; Ito, E.: "Design Principles and Algorithms for Air Traffic Arrival Scheduling," NASA/TP-2014-218302, May 2014.

[25] Isaacson, D.R.; Sadovsky, A.V.; Davis, D.: "Tactical Scheduling for Precision Air Traffic Operations: Past Research and Current Problems," *Journal of Aerospace Information Systems*, Vol. 11, No. 4, April 2014.

[26] "Air Traffic Control," FAA Order JO 7110.65Y, Federal Aviation Administration, Dept. of Transportation, Washington, DC., 2020.

[27] Alligier, R.; Durand, N.; Alligier, G.: "Efficient Conflict Detection for Conflict Resolution," *8th International Conference on Research in Air Transportation* (ICRAT 2018) Castelldefels, Spain, Jun 2018.

[28] Allignol, C.; Barnier, N.; Durand, N.; Gondran, A.; Wang, R.: "Large-Scale 3D En-Route Conflict Resolution," *USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)*, 2017.

[29] Zhang, Y.; Satapathy, G.; Manikonda, V.; Nigam, N.: "KTG: A Fast-time Kinematic Trajectory Generator for Modeling and Simulation of ATM Automation Concepts and NAS-wide System Level Analysis," *AIAA Modeling and Simulation Technologies Conf.*, Toronto, Ontario Canada, 2-5 Aug 2010. https://doi.org/10.2514/6.2010-8365

[30] Eurocontrol Experimental Centre, "Base of Aircraft Data (BADA) Aircraft Performance Modelling Report," EEC Technical/Scientific Report No. 2009-009, 2009.

[31] Abstract Syntax Notation: http://www.itu.int/en/ITU-T/asn1

[32] Efficient XML Interchange (EXI): https://www.w3.org/TR/exi/

A Trajectory Specification Language

The Trajectory Specification (TS) concept will require a standard language to represent and communicate the specifications between aircraft and the ATC system. The underlying data communication technology that would be used is outside the scope of this paper, but a likely candidate is the developing Internet Protocol Suite (IPS) for Air Traffic Services (ATS). A proposed language called the Trajectory Specification Language (TSL) has been developed [14] based on XML and will be presented in this section. TSL can be used to downlink trajectory requests and to uplink trajectory assignments, and a conforming FMS will be programmed to understand the language and to keep the flight in conformance.

The use of XML is not required for TSL, but XML was chosen to show an example of how the language could be implemented. The structure and form of an XML document can be formally described by a Document Type Definition (DTD) or an XML Schema Definition (XSD), but the objective here is not to formally define an XML document. The objective is to show how the format should be structured and what information it should contain. Example XML elements will be presented and explained to provide a high-level design that can be used to develop a formal Schema.

The XML format can be converted to binary for operational use, if necessary, using Efficient XML Interchange (EXI) [32] or Abstract Syntax Notation (ASN.1) [31]. EXI is a binary compression method that is significantly more efficient for XML than standard data compression methods such as zip or gzip, both in terms of size and conversion speed. Alternatively, the equivalent of an XML schema can be defined using ASN.1, an industry standard for defining platform-independent binary data formats. ASN.1 includes XML Encoding Rules (XER) to facilitate equivalent binary and XML data formats and conversion between the two.

An XML document consists of a hierarchy of elements, each of which can contain subelements and/or attributes. Consider, for example, the following XML element:

```
<note from="building manager" to="occupants">
  <subject>electrical maintenance</subject>
  <body>Power in bldg 210 will be out ...</body>
  </note>
```

The main delimiters are angle brackets, which enclose the opening and closing tags of each element or subelement. The example shows an element called `note`, which has attributes `from` and `to`, and which has subelements `subject` and `body`. Attributes are specified in the opening tag of an element, and the closing tag contains the element name preceded by a forward slash. Attribute values should be in quotes as shown. Attributes are supposed to contain metadata, but the distinction between data and metadata is not always obvious. (The convention in this paper is to indent the closing tag one level more than the opening tag to better preserve the logical structure.)

A DTD or an XSD can restrict allowed values to a specified discrete set. It can also restrict attributes and elements to specified data types, such as text, boolean, integer, or decimal number. Other structural and ordering restrictions can also be imposed, but they will not be discussed here.

Table 3: Default Physical Units

quantity	unit
time	second (sec)
horizontal length and position	nautical mile (nmi)
vertical length and altitude	foot (ft)
angle	degree (deg)

The XML elements to be discussed below are serializations of the main classes in the prototype software. As a serialization of the software classes, the TSL has also been useful for software development. Fast-time simulation testing on a full day of traffic can take hours to run, and if a conflict is not resolved properly, the TSL can be used to capture the detailed traffic state at the time of the conflict. That capability allows the developer to restart the simulation at the desired state rather than having to wait, possibly for hours, for the same traffic situation to develop in order to see the effect of each software change on a particular conflict.

The design of TSL involves many choices on which reasonable people can disagree, particularly involving the degree of low-level structure. Should each coordinate of a position be a subelement, for example, or should the coordinates simply be concatenated together into a comma-separated list of numbers? XML purists will argue that more structure is better because it reduces the chance of an error. For most XML applications that may be true, but it is not necessarily true for this safety-critical application because the barrier to entry is very high. FMS certification is a rigorous process with extensive testing, and errors involving the order of coordinates would obviously be detected and corrected very early. Hence, for simplicity, the approach taken in this paper is to avoid some low-level structure, but that can easily be changed if a future standards committee decides to do so.

Standard aviation units are used as shown in Table 3. Time is given in seconds (sec), horizontal distance or length is in nautical miles (nmi), vertical length or altitude is in feet (ft), and angles are in degrees (deg). Allowing alternate units can enhance flexibility but also increases the risk of error; whether or not to allow them will be deferred to a future standards committee. If alternate units are not allowed, the unit attributes shown in the examples to follow will not be needed. If alternate units are allowed, they should be from a very small set of options to minimize the risk of error because all users need to be able to recognize and convert them. The allowed units for altitude could be feet or meters, for example, and the allowed units for distance could be nautical miles or kilometers.

A.1 Trajectory

The specification for a trajectory is represented by the `traj` element, the structure of which is shown in XML sample 1. The `name` attribute could be the call sign or any other unique and appropriate identifier. The `time` attribute is the assignment or request time in seconds (unix time). The optional `assign` attribute is a boolean that defaults to false and is true if the trajectory is an assignment (as opposed to a request or a resolution maneuver candidate). A possible alternative to the boolean `assign` attribute could be a `status` attribute that is limited to a small set of enumerated values, such as `ASSIGN`, `REQUEST`, and `CANDIDATE`. The `GUFI` attribute holds the Globally Unique Flight Identifier, which is required for all ATC flight data transactions.

The subelements of the `traj` element are `flight` (flight information), `route`, `refTraj` (reference trajectory), `altTols` (altitude tolerances), and `alongTols` (along-track tolerances). As explained earlier, these elements correspond to the Scala classes that were developed to implement the concept. Although not shown, each subelement could have its own optional `name` attribute for reference (which should all be the same).

XML Sample 1: Trajectory structure

```
<traj name="AAL123" time="1378478953" assign="true" GUFI="...">
    <flight name="AAL123" info="B738 Large DFW 18R Arr"/>
    <route> ... </route>
    <refTraj> ... </refTraj>
    <altTols> ... </altTols>
    <alongTols> ... </alongTols>
    </traj>
```

XML Sample 2: Flight information

```
<flight name="AAL123">
    <aircraft>B738</aircraft>
    <weightClass>Large</weightClass>
    <airport>DFW</airport>
    <runway>18R</runway>
    <type>Arrival</type>
    </flight>
```

The `flight` subelement contains basic flight information in its `info` attribute, including the aircraft type code (B738 in this example), the weight class (Large), the airport code (DFW), the runway (18R), and the flight type (`Arr` for arrival or `Dep` for departure). This information could be broken down into individual subelements as shown in XML sample 2, which is more explicit and can be parsed using standard XML tools. However, the simpler form reads naturally and is trivial to parse. A standard equipment code could be appended to the aircraft type code if necessary (after a slash, as usual). The other subelements of the trajectory (`traj`) element are explained in the following subsections.

A.2 Route

The route is the planview of the trajectory, consisting of straight segments and turn segments. The structure of the `route` element is shown in XML sample 3. The basic flight information (`flight`) from the `traj` element is repeated for reference because the route depends on that information. The reference along-track starting distance (`startDist`) is arbitrary and was set to -53.031923 nmi to make the along-track distance zero at the end of the trajectory (at the runway threshold). The cross-track tolerance (`crossTol`) is set to 0.6 nmi, which is equivalent to RNP 0.3.

The waypoint list (`waypts`) of the `route` element has a `type` attribute, which can be `local` or `global`, depending on whether the waypoint locations are given in terms of a locally level map projection or global geodetic coordinates (latitude and longitude). If a local map projection is used, the `frame` attribute specifies the name of the frame that is used, which is D10 in this case, a predefined local frame for the D10 TRACON airspace. A predefined frame (defined by its projection type and tangency point) would be published for each major TRACON airspace and would never change. If a global frame is used, it would default to WGS-84 (or whatever a standards committee deems appropriate). As explained earlier, the unit attributes could be optional and default to `nmi`

XML Sample 3: Route element

```
<route name="1366">
    <flight name="1366" info="B738 Large DFW 18R Arr"/>
    <startDist unit="nmi">-53.031923</startDist>
    <crossTol unit="nmi">0.6</crossTol>
    <waypts type="local" frame="D10" unit="nmi">
        <waypt name="HOWDY"> 27.917174, -31.050092 <rad>2.000</rad></waypt>
        <waypt> 27.235636, -27.108529 <rad>3.000</rad></waypt>
        ...
        </waypts>
    </route>
```

XML Sample 4: Cross-track tolerance element

```
<crossTol unit="nmi">0.6 / 23.5: 2.0</crossTol>
```

if not given.

The cross-track tolerance should rarely change during a flight except possibly when entering or exiting terminal airspace. The `crossTol` subelement of the `route` element could represent such a change by appending data to the first cross-track tolerance as shown in XML sample 4. In this example, the initial cross-track tolerance of 0.6 nmi is followed after a slash by `23.5: 2.0`, which means that the cross-track tolerance changes to 2.0 nmi at the along-track distance of 23.5 nmi. Changes in cross-track tolerances are instantaneous (discontinuous) at the change point (rather than piecewise linear as for the vertical and along-track tolerances to be discussed later). Additional changes further along the route could follow similarly, if necessary. Additional structure could be added here but is probably not necessary because, as explained earlier, the FMS certification process is so rigorous that basic parsing errors would not survive it.

The individual waypoints are each specified by the `waypt` element. Each waypoint can have an optional `name` attribute, which is provided for reference only and has no effect on the resulting route. The first waypoint in the example is called "`HOWDY`," and the coordinates of its position are 27.917174 nmi and -31.050092 nmi. If geodetic coordinates are used, the coordinates would be latitude and longitude in degrees. The `rad` subelement represents the turn radius at the waypoint, which is 2.0 nmi for the first waypoint in this case. A turn radius is required for all but the first and last waypoints. All turns are tangent arc "flyby" turns, and if a specified turn radius cannot be accommodated for the given the leg lengths, the route will be rejected as geometrically invalid. The use of six digits after the decimal place provides a high resolution that is nearly equivalent to binary. More digits would be needed for geodetic coordinates to achieve the same level of resolution. The ellipses indicate that several waypoints have been omitted for brevity in this example.

XML Sample 5: Reference trajectory element

```
<refTraj name="1366">
    <dt unit="sec">5.000</dt>
    <refTime unit="sec">1378478942.200</refTime>
    <points type="local" frame="D10" units="sec,nmi,ft">
        <pt>0.000,27.905328,-30.981547,9986</pt>
        <pt>5.000,27.837196,-30.587082,9982</pt>
        <pt>10.000,27.768812,-30.192529,9981</pt>
        ...
        <pt>750.000,7.024273,-2.354429,481</pt>
        </points>
    </refTraj>
```

A.3 Reference Trajectory

The reference trajectory is the ideal predicted trajectory that would be flown in the absence of any wind modeling errors or other modeling errors. It is represented by the `refTraj` element, the structure of which is shown in XML sample 5. The time step (`dt`) in this example is 5.0 sec, and the reference time (`refTime`) is 1378478942.2 sec. The reference time is simply the starting time of the reference trajectory and is used to save space and enhance readability.

Like the `waypts` subelement of the `route` element discussed above, the points list (`points`) has a `type` attribute, which can be `local` or `global`, depending on whether the points are given in terms of a locally level map projection or global geodetic coordinates. If a local map projection is used, the `frame` attribute specifies the name of the frame that is used, which is again `D10` in this case. The `units` attributes is shown as "`sec,nmi,ft`", meaning that the time is in seconds, horizontal position is in nautical miles, and altitude is in feet.

Each point (`pt`) in the list of points has a time stamp (relative to the reference time), x and y coordinates of position, and an altitude, all separated by commas and (optional) spaces. The first point has a time of 0.0 sec, x and y coordinates of 27.905328 nmi and -30.981547 nmi, and an altitude of 9986 ft. Again, additional subelement structure could be added here, but the FMS certification process is so rigorous that basic parsing errors would not survive it. As before, the use of six digits after the decimal place provides a high resolution almost equivalent to binary. Again, if geodetic coordinates are used, the horizontal coordinates would be latitude and longitude in degrees, and more digits will be required for the same level of resolution. The ellipses indicate that many points have been omitted for brevity in this example.

An alternate form of the point (`pt`) element is also possible. Rather than specifying position in terms of locally level or geodetic coordinates, it could be specified in terms of along-track distance. In that case, each point would contain the time stamp, the along-track distance, and the altitude. This alternate form has two advantages. Firstly, it reduces the number of position coordinates from two to one, significantly reducing storage space and transmission bandwidth requirements. Secondly, it eliminates the possibility that the specified point could be off the route (because the cross-track coordinate would always be zero by definition). The disadvantage, perhaps insignificant, is that the actual position on the surface of the earth is not obvious, and the potential for error may be slightly higher.

Note that the points of the reference trajectory need not be uniformly spaced in time as shown in the example. They could be spaced further apart in steady-state flight, for example, but

XML Sample 6: Altitude tolerances element

```
<altTols units="nmi, ft">
    <tol> 0: -500, 500 </tol>
    <tol> 40: -1000, 1000 </tol>
    <taper unit="deg">2.5</taper>
    </altTols>
```

they should be spaced 5 seconds or less in turns and altitude transients. They will be converted internally by the TS software to a uniformly spaced sequence of time steps specified by the `dt` element. (Uniform time steps are a key to efficient computation because they allow fast access of trajectory data points as a function of time by array indexing and interpolation). The reference trajectory typically takes up approximately 80-90% of the overall storage space for the trajectory specification.

A.4 Altitude Tolerances

Figure 2 shows an example of a side view of the stationary, rectangular tube in which an aircraft is required to fly. The TS software automatically detects the level segments and applies the default tolerance of ±200 ft in those segments. It also applies the tapered transitions between the level and non-level segments and cuts off the altitude overshoots that would otherwise precede or follow a level segment. The tapered transitions are shown at a slope of 2.5 deg but could be varied slightly. The smaller the taper angle is, the more airspace is reserved. The taper angle should be slightly less than the climb or descent angle to allow enough room for a normal leveloff or start of climb or descent.

The tolerances for level flight are not explicitly specified but are assumed to be consistent with the current "altitude rounding" rule: if an aircraft is in level flight within ±200 ft of its cleared altitude, it is considered to be at its cleared altitude for purposes of separation requirements. (Without this rule, many nuisance alerts would occur due to small altitude deviations in cruise.) The taper angle for transition to and from level flight is specified in the optional `taper` subelement, and the default value is 2.5 deg if not specified. The tolerances for level flight (including the tapered transitions to and from level flight) override the explicitly specified tolerances discussed below, which apply in climb and descent.

The altitude tolerances are specified in the `altTols` element, the structure of which is shown in XML sample 6. The piecewise linear tolerances are specified as a list of tolerance (`tol`) points, where each point contains the tolerance values separated by commas and preceded by the along-track distance at which it applies followed by a colon (spaces are optional). The first `tol` subelement shows vertical tolerances of ±500 ft at an along-track distance of 0 nmi. The lower tolerance is given first as a negative value, followed by the upper tolerance. By definition, those tolerances apply for any along-track distance less than the first specified distance of 0 nmi. The second `tol` point shows that the tolerances increase linearly to ±1000 ft at an along-track distance of 40 nmi and, because that is the last `tol` point, remain constant beyond that distance by definition. (If there is only one `tol` point, then the tolerances are constant, and the along-track distance for that point is irrelevant.) Although the lower and upper tolerances are equal in this example, in general they can be different.

XML Sample 7: Along-Track tolerances element

```
<alongTols unit="nmi">
    <tol> -53.0319: -1.0, 1.0 </tol>
    <tol> -10: -0.2, 0.2 </tol>
    </alongTols>
```

A.5 Along-Track Tolerances

The along-track tolerances are specified in the `alongTols` element, the structure of which is shown in XML sample 7. Again, the piecewise linear tolerances are specified as a list of tolerance (`tol`) points, where each point contains the tolerance values separated by commas and preceded by the along-track distance at which it applies followed by a colon (spaces are optional). The first `tol` subelement below shows along-track tolerances of ±1 nmi at an along-track distance of -53.0319 nmi. The back tolerance is given first as a negative value, followed by the front tolerance.

The starting distance was chosen in this example to be same as the `startDist` of the route, which was set to make the along-track distance zero at the runway threshold. By definition, those tolerances apply for any along-track distance less than that first specified distance. The second `tol` point shows that the tolerances decrease linearly to ±0.2 nmi at an along-track distance of -10 nmi and, because that is the last `tol` point, remain constant beyond that distance by definition. (Again, if there is only one `tol` point, then the tolerances are constant, and the along-track distance for that point is irrelevant.) Such a decreasing along-track tolerance would be typical during an arrival rush when throughput is critical. Again, although the front and back tolerances are equal in this example, they can be different.

The trajectory tolerances are not required in a downlinked trajectory request from an aircraft because they will be assigned by ATC based on published aircraft navigational capabilities and the current traffic situation. Allowing the pilot or FMS to specify arbitrary tolerances would not make sense because the only incentive would be to request the largest possible tolerances. If tolerances are allowed in the downlinked trajectory request, they should be the tightest tolerances that the aircraft has been determined to be capable of conforming to. A database should be established so that the ATC system can lookup the navigational capabilities of each aircraft based on the aircraft type and equipage code.

A.6 Trajectory Updates

Trajectory updates can often be done more efficiently without sending an entire new trajectory. A simple time shift to adjust for accumulated wind errors, for example, can be specified by the time shift rather than an entire new trajectory. An example of how that could be specified is shown in XML sample 8. As before, the time attribute represents the assignment (update) time, and the timeshift element is the time by which the reference trajectory is shifted, which is -8 sec in this case. When received by the aircraft, the FMS would generate an entire new trajectory representation for its own use, but that trajectory need not take up communication bandwidth.

As another example of a trajectory update, consider changing the tolerances without changing the route or the reference trajectory. That could be done by using the `traj` element discussed earlier and simply omitting the route and the reference trajectory if they are unchanged.

XML Sample 8: Time shifting example

```
<traj name="1366" time="1378478953" assign="true" GUFI="...">
  <timeshift unit="sec">-8.000</timeshift>
  </traj>
```

A.7 Data Communication Requirements

As mentioned earlier, the underlying data communication technology that would be used for TSL is outside the scope of this paper, but a likely candidate is the developing Internet Protocol Suite (IPS) for Air Traffic Services. This new technology is expected to significantly increase the bandwidth available for applications such as TSL.

To quantify the data transfer requirements of the proposed language, a typical terminal trajectory was selected at random and serialized using the language. Trajectories through the terminal airspace typically range from 10 to 15 minutes in length, and the selected trajectory was approximately 15 minutes long. The serialization was found to contain approximately 10.1 kilobytes before compression and 2.6 kilobytes after compression with gzip, for a compression ratio of approximately 3.9. EXI is better than gzip for compressing XML but was not used here. The reference trajectory takes up approximately 84% of the storage space.

Enroute trajectories are usually longer than terminal trajectories, up to several hours in length, so the data quantities could be larger, but the entire reference trajectory need not be specified if it is over approximately an hour in length. Enroute trajectories will be updated occasionally to adjust for the accumulated effect of wind errors and to resolve conflicts as they arise within the conflict-free time horizon of approximately 20 to 30 minutes. These updates will be needed perhaps once every 10 to 20 minutes or so, but many of them should be simple time shifts. Since these updates are necessary anyway, only the first 30 minutes to an hour of the reference trajectory needs to be specified at any given time. Moreover, enroute trajectories also tend to have more and longer periods of steady-state (straight and level) flight. Steady-state segments can be accurately represented with longer time steps of 30 sec or more, compared to 5 sec or less for turns and other non-steady segments, reducing the amount of data required.

To put these data quantities into perspective, consider the amount of data involved in streaming or downloading music to a mobile device. A song of 4 minutes in length at a typical resolution of 32 kbps requires roughly 1 MB of data to be transmitted. That is enough for roughly 100 uncompressed terminal airspace trajectories or 400 such trajectories after compression. Those numbers should be reduced by roughly an order of magnitude for longer enroute trajectories. If these data rates are not acceptable, or if the XML processing is too slow, a binary equivalent of the proposed XML format based on EXI or ASN.1 can be used as discussed earlier.

List of Figures

Many supplemental plots for this paper can be viewed at `http://RussP.us/plots`
The Scala software that was used for this paper is available at `https://github.com/RussP`

About the author

Russ Paielli (Russ.Paielli@gmail.com) is an aerospace engineer in the Flight Trajectory Dynamics and Control Branch of the Airspace Systems Division at NASA Ames Research Center. He has a BS degree in Mechanical Engineering from Oakland University (in Michigan) and an MS degree in Aeronautics and Astronautics from Stanford University. He has been with NASA since 1982, where he has worked on flight control theory, precision navigation and landing, and air traffic control automation, among other things. A list of his technical publications is available at `http://RussP.us/publist.htm`